

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号  
特表2002-517034  
(P2002-517034A)

(43) 公表日 平成14年6月11日 (2002.6.11)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード(参考)
G 0 6 F 9/38	3 7 0	G 0 6 F 9/38	3 7 0 C 5 B 0 1 3
15/16	6 2 0	15/16	6 2 0 G 5 B 0 4 5

審査請求 未請求 予備審査請求 有 (全 73 頁)

(21) 出願番号 特願2000-551315(P2000-551315)  
 (86) (22) 出願日 平成11年1月25日(1999.1.25)  
 (85) 翻訳文提出日 平成12年11月24日(2000.11.24)  
 (86) 国際出願番号 PCT/US99/01456  
 (87) 国際公開番号 WO99/61981  
 (87) 国際公開日 平成11年12月2日(1999.12.2)  
 (31) 優先権主張番号 09/085,187  
 (32) 優先日 平成10年5月26日(1998.5.26)  
 (33) 優先権主張国 米国(US)  
 (81) 指定国 EP(AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), JP, KR

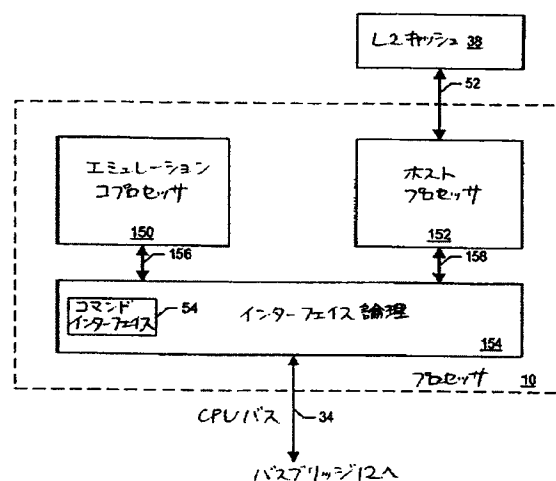
(71) 出願人 アドバンスド・マイクロ・ディバイシズ・  
インコーポレイテッド  
ADVANCED MICRO DEVI  
CES INCORPORATED  
アメリカ合衆国、94088-3453 カリフォルニア州、サニイベイ、ビィ・オー・ボックス・3453、ワン・エイ・エム・ディ・ブレイス、メイル・ストップ・68 (番地なし)  
 (74) 代理人 弁理士 深見 久郎 (外5名)

最終頁に続く

(54) 【発明の名称】 エミュレーションコプロセッサ

## (57) 【要約】

ホストプロセッサ(48、152)およびエミュレーションコプロセッサ(50、150)を採用するコンピュータシステム(5)である。ホストプロセッサ(48、152)は、ホスト命令セットアーキテクチャによって定義される命令を実行するよう構成されるハードウェアを含み、エミュレーションコプロセッサ(50、150)は、ホスト命令セットアーキテクチャと異なる命令セットアーキテクチャ(「外部命令セットアーキテクチャ」)によって定義される命令を実行するよう構成されるハードウェアを含む。ホストプロセッサコア(48、152)は、オペレーティングシステムコードおよび、ホスト命令セットアーキテクチャ内にコード化されるアプリケーションプログラムを実行する。外部アプリケーションプログラムが起動されると、ホストプロセッサコア(50、150)は、エミュレーションコプロセッサコア(48、152)と交信して、エミュレーションコプロセッサコア(48、152)が外部アプリケーションプログラムを実行するようにする。したがって、外部命令セットアーキテクチャに従ってコード化されたアプ



**【特許請求の範囲】**

**【請求項1】** コンピュータシステムのための装置であって、

第1の命令セットアーキテクチャによって定義される第1の命令を実行するよう構成される第1のプロセッサ（48、152）を含み、前記コンピュータシステム（5）によって採用されるオペレーティングシステム（84）は、前記第1の命令を用いてコード化され、さらに、

前記第1のプロセッサ（48、152）に結合される第2のプロセッサ（50、150）を含み、前記第2のプロセッサ（50、150）は、前記第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャによって定義される第2の命令を実行するよう構成され、前記オペレーティングシステム（84）内で実行するよう設計されるアプリケーションプログラム（82）は、前記第2の命令を用いてコード化され、

前記第2のプロセッサ（50、150）は、前記アプリケーションプログラム（82）を実行するよう構成され、前記第1のプロセッサ（48、152）は、前記オペレーティングシステム（84）を実行するよう構成され、前記第2のプロセッサ（50、150）は、前記アプリケーションプログラム（82）のためのオペレーティングシステムルーチン（94）の使用を検出すると前記第1のプロセッサ（48、152）と通信するよう構成される、コンピュータシステムのための装置。

**【請求項2】** 前記第1のプロセッサ（48）および前記第2のプロセッサ（50）は、1つ以上のキャッシュ（44、46）に結合され、前記第1のプロセッサ（48）および前記第2のプロセッサは、前記1つ以上のキャッシュ（44、46）を共有するよう構成される、請求項1に記載の装置。

**【請求項3】** 前記第1のプロセッサ（48）および前記第2のプロセッサ（50）は、1つ以上のメモリ管理ユニット（42）に結合され、前記第1のプロセッサ（48）および前記第2のプロセッサ（50）は、前記メモリ管理ユニット（42）を共有するよう構成される、請求項2に記載の装置。

**【請求項4】** 前記第1のプロセッサ（48）および前記第2のプロセッサ（50）は、単一の半導体基板（10）上に集積される、請求項3に記載の装置

。

【請求項5】 前記第1のプロセッサ(152)および前記第2のプロセッサ(150)は、バスブリッジ(12)に結合され、前記第1のプロセッサ(152)は、CPUバス(34)を介して前記バスブリッジ(12)に結合され、前記第2のプロセッサ(150)は、前記CPUバス(34)と異なった信号法を有する周辺バス(24)を介して前記バスブリッジ(12)に結合される、請求項1に記載の装置。

【請求項6】 前記第2のプロセッサ(50、150)は、前記第2の命令をデコードするよう構成されるハードウェアデコーダ(70)を含む、請求項1に記載の装置。

【請求項7】 前記第1のプロセッサ(48、152)および前記第2のプロセッサ(50、150)は、予め定められた制御プロトコルを介して交信するよう構成される、請求項1に記載の装置。

【請求項8】 前記制御プロトコルは、前記第1のプロセッサ(48、152)と前記第2のプロセッサ(50、150)との間でやり取りされるメッセージを含む、請求項7に記載の装置。

【請求項9】 前記メッセージは、前記コンピュータシステム(5)内のメモリ(14)を介してやり取りされる、請求項8に記載の装置。

【請求項10】 前記メッセージは、前記第1のプロセッサ(48、152)と前記第2のプロセッサ(50、150)との間の専用通信チャネル(54)を介してやり取りされる、請求項8に記載の装置。

【請求項11】 異種のマルチプロセッシングシステムであって、  
第1の命令セットアーキテクチャによって定義される第1の命令を実行するよう構成される第1のプロセッサ(48、152)と、  
前記第1のプロセッサ(48、152)に結合される第2のプロセッサ(50、150)とを含み、前記第2のプロセッサ(50、150)は、前記第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャによって定義される第2の命令を実行するよう構成され、さらに、  
前記第1の命令を用いてコード化されるオペレーティングシステム(84)と

、  
前記第2の命令セットを用いてコード化されかつ前記オペレーティングシステム(84)内で実行するよう設計されるアプリケーションプログラム(82)とを含み、

前記第2のプロセッサ(50、150)は、前記アプリケーションプログラム(82)を実行するよう構成され、前記第1のプロセッサ(48、152)は、前記アプリケーションプログラム(82)と関連しないプロセス(80)を同時に実行するよう構成される、異種のマルチプロセッシングシステム。

【請求項12】 前記第2のプロセッサ(50、150)は、実行中に前記アプリケーションプログラム(82)による前記オペレーティングシステム(84)内のオペレーティングシステムルーチン(94)の使用を検出するよう構成される、請求項11に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項13】 前記第2のプロセッサ(50、150)は、特定の違法操作コードを実行することによって前記使用を検出するよう構成される、請求項12に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項14】 前記第2のプロセッサ(50、150)は、前記使用を検出すると前記第1のプロセッサ(48、152)と交信するよう構成される、請求項12に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項15】 前記第1のプロセッサ(48、152)は、前記第2のプロセッサ(50、150)からコンテキスト情報を要求し、前記オペレーティングシステムルーチン(94)を実行し、前記第2のプロセッサ(50、150)との交信によって前記アプリケーションプログラム(82)の制御を前記第2のプロセッサ(50、150)に戻すよう構成される、請求項14に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項16】 前記アプリケーションプログラム(82)と関連しない前記プロセス(80)は、第2のアプリケーションプログラムを含む、請求項11に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項17】 第1の命令セットアーキテクチャからの命令を用いてコー

ド化され、前記第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャからの命令を用いてコード化されるオペレーティングシステム（84）内で実行するよう設計されるアプリケーションプログラム（82）を実行するための方法であって、

前記アプリケーションプログラム（82）が起動されていることを検出するステップを含み、前記検出は、前記第2の命令セットアーキテクチャからの命令を実行するよう構成される第1のプロセッサ（48、152）上で実行する前記オペレーティングシステム（84）によって行なわれ、さらに、

前記第1の命令セットアーキテクチャからの命令を実行するよう構成される第2のプロセッサ（50、150）内の前記アプリケーションプログラム（82）のためのコンテキストを確立するステップと、

前記第2のプロセッサ（50、150）上で前記アプリケーションプログラム（82）を実行するステップとを含む、方法。

【請求項18】 前記オペレーティングシステム（84）内のオペレーティングシステムルーチン（94）への前記アプリケーションプログラム（82）内の遷移を検出するステップをさらに含む、請求項17に記載の方法。

【請求項19】 前記第1のプロセッサ（48、152）上で前記オペレーティングシステムルーチン（94）を実行するステップをさらに含む、請求項18に記載の方法。

【請求項20】 前記オペレーティングシステムルーチン（94）を前記実行した後に前記第2のプロセッサ（50、150）上で実行する前記アプリケーションプログラム（82）に戻るステップをさらに含む、請求項19に記載の方法。

**【発明の詳細な説明】****【0001】****【発明の背景】****1. 技術分野**

この発明は、コンピュータシステムのためのプロセッサの分野に関し、より特定のには、コンピュータシステム内の複数の命令セットアーキテクチャをサポートすることに関する。

**【0002】****2. 背景技術**

コンピュータシステムは、多くの環境において重要な生産性ツールとなってきた。仕事のほぼすべてのラインが、その仕事の中心となる多くのタスクを行なうためにコンピュータシステムから恩恵を受けている。たとえば、管理専門職は、ビジネス上重要なデータのデータベースを管理し、書類を作成し管理するなどのためにコンピュータシステムを使用する。技術専門職は、製品の研究、設計および検証のためにコンピュータシステムを使用する。製造および流通センターは、製造機械を制御し、在庫管理のために製造プロセスを通る製品を追跡し、卸売／小売センターへの流通製品を管理するためにコンピュータシステムを使用する。以上のすべてが、Eメール、インターネット、イントラネットなどを介する通信のためにもコンピュータシステムを使用するであろう。家計管理、通信および娯楽を含む、コンピュータシステムの家庭での使用もまた多い。コンピュータシステムのその他の多くの使用が存在する。

**【0003】**

以上が示すように、コンピュータシステムのための多様な組の使用が開発されてきた。一般的には、これらの使用は、コンピュータシステムに設けられたオペレーティングシステム下で実行するよう設計されるさまざまなアプリケーションプログラムによってサポートされる。オペレーティングシステムは、アプリケーションプログラムとコンピュータシステムハードウェアとの間にインターフェイスを与える。各コンピュータシステムは、ハードウェア構成（たとえば、メモリの量、入力／出力（I/O）デバイスの数およびタイプなど）においてさまざま

な差異を有するであろう。オペレーティングシステムは、ハードウェアの差異からアプリケーションプログラムを切り離す。したがって、アプリケーションプログラムはしばしば、アプリケーションプログラムが実行すべき正確なハードウェア構成に関係なしに設計され得る。加えて、オペレーティングシステムは、多くの異なったタイプのアプリケーションプログラムが必要とするであろうさまざまなローレベルのサービスを与え、アプリケーションプログラムの内側にこれらのサービスをプログラミングする代わりにアプリケーションプログラムがオペレーティングシステムサービスに依存することを可能にする。一般的に、オペレーティングシステムは、タスクのスケジューリング（たとえば、同時に実行可能である異なったアプリケーションプログラム）、I/Oデバイスおよびメモリなどのシステム資源の管理および割当て、エラー処理（たとえば誤って動作するアプリケーションプログラム）などを与える。オペレーティングシステムの例は、中でも、ウィンドウズ(Windows)オペレーティングシステム（ウィンドウズ95およびウィンドウズNTを含む）、UNIX、DOSおよびMAC-OSである。反対に、アプリケーションプログラムは、特定のユーザタスクを達成するために特定のユーザ機能を与える。ワードプロセッサ、スプレッドシート、グラフィックスデザインプログラム、在庫管理プログラムなどが、アプリケーションプログラムの例である。

#### 【0004】

したがって、アプリケーションプログラムは典型的には、特定のオペレーティングシステム上で動作するよう設計される。オペレーティングシステムから利用可能であるサービス（「オペレーティングシステムルーチン」）が、アプリケーションプログラムによって任意で使用される。加えて、アプリケーションプログラムは、オペレーティングシステムの要件に従う。

#### 【0005】

オペレーティングシステムが典型的にアプリケーションプログラムを切り離さないハードウェア特徴の1つは、コンピュータシステム内のプロセッサの命令セットアーキテクチャである。一般的に、命令セットアーキテクチャは、プロセッサ上で実行する命令および、命令によって直接使用されるプロセッサ資源（レジ

スタなど)を定義する。アプリケーションプログラムは典型的には、命令セットアーキテクチャによって定義される命令の組に従い、このためオペレーティングシステムは、コンピュータシステムハードウェアのこの特徴からアプリケーションプログラムを切り離さない。

#### 【0006】

上述したように、コンピュータシステムは、広範なカスタマにとって有用であるように多数の異なったタイプのアプリケーションプログラムをサポートしなければならない。新しく開発された命令セットアーキテクチャを採用するプロセッサは、アプリケーション開発者を、新しい命令セットアーキテクチャのために設計されたアプリケーションを開発するという困難な課題に直面させる。しかしながら、アプリケーションプログラムなしでは、命令セットアーキテクチャおよびそのために設計されたプロセッサはしばしば、せいぜい限られた市場の支持しか達成しないであろう。

#### 【0007】

多数のアプリケーションプログラムと、各アプリケーションプログラムを新しい命令セットアーキテクチャに「ポート」するために必要とされる時間および労力とのために、新しい命令セットアーキテクチャを用いてアプリケーションプログラムを再び作ることは、困難かつ時間のかかることである。さらに、多くのアプリケーションプログラムについてのソースコードは、ポートを実行しようと望んでいるものにとって利用不可能であろう。他方で、オペレーティングシステムは(特に広範に受入れられているものについては)数がより少なく、さまざまな命令セットアーキテクチャにポートされるであろう。たとえば、ウィンドウズNTは、x86アーキテクチャに加えて、デジタル・イクイップメント社(Digital Equipment Corporation)によって開発されたアルファ(Alpha)アーキテクチャ、IBMおよびモトローラ(Motorola)によって開発されたパワーPC(Power PC)アーキテクチャおよびMIPSアーキテクチャをサポートする。

#### 【0008】

大きいアプリケーションベースを提供し、これによってより多くのアプリケーションプログラムが開発されるように導くであろう市場支持を生み出すために、



新しく開発された命令セットアーキテクチャを採用するプロセッサに基づくコンピュータシステムは、異なった命令セットアーキテクチャにコード化されたアプリケーションをサポートするであろう。ここでは、コンピュータシステム内のプロセッサによって採用される命令セットアーキテクチャによって定義される命令を用いるコードは、「ネイティブ」または「ホスト」と呼ばれ、異なった命令セットアーキテクチャによって定義される命令を用いるコードは、「ノンネイティブ」または「外部 (foreign)」と呼ばれる。

#### 【0009】

(IA-32またはAPXとも呼ばれる) x86アーキテクチャは、コンピュータ史上最大のアプリケーションプログラムベースの1つである。これらのプログラムの大部分が、ウィンドウズオペレーティングシステム下で実行するように開発されている。ウィンドウズおよびx86アプリケーションプログラムは、ここに例として定期的に用いられるが、ここに開示される技術およびハードウェアは、この命令セットアーキテクチャおよびオペレーティングシステムに限られるものではない。いかなるオペレーティングシステムおよび命令セットアーキテクチャが使用されてもよい。

#### 【0010】

ホストプロセッサが非x86である新しいコンピュータシステムは、ウィンドウズオペレーティングシステム下で実行するx86 (すなわち外部) アプリケーションプログラムに対するサポートを提供し得るが、アプリケーションプログラムは非x86ホストプロセッサのために開発されている。コンピュータシステム内で外部アプリケーションをサポートするために用いられてきた2つの方法は、ソフトウェアエミュレーションおよびバイナリ変換である。ソフトウェアエミュレーションは一般的に、命令が選択され実行されるとアプリケーションプログラム内の各命令を読出すことと、ホストアーキテクチャ内の等価の命令シーケンスを実行することとを含む。バイナリ変換は一般的には、プログラムを実行するより前に、アプリケーションプログラム内の各命令を等価の命令シーケンスに変換し、次に変換されたプログラムシーケンスを実行することを伴う。

#### 【0011】

残念ながら、各外部命令は、プログラムの実行中に調べられるので、ソフトウェアエミュレーションは、外部命令セットアーキテクチャを採用するコンピュータシステム上で達成可能なものよりも、アプリケーションプログラムの性能が大きく低減されている。さらに、エミュレーションプログラムを記憶しデータ構造をサポートするために、より多くのメモリがアプリケーションプログラムを実行するために必要とされる。アプリケーションプログラムが、リアルタイム特徴（たとえばオーディオおよびビデオ）を含む場合、これらの特徴は、過剰な実行時間のために不良に動作するであろう。さらになお、命令セットアーキテクチャのプロセッサ実現化例はしばしば、ソフトウェアエミュレータによってモデリングされなければならないさまざまなドキュメントされていない特徴（既知および未知）を含む。さらに、（x86浮動小数点レジスタスタックなどの）複雑なハードウェア特徴は、ソフトウェアエミュレータ内で正確にモデリングするのが困難である。

#### 【0012】

バイナリ変換もまた、いくつかの欠点がある。バイナリ変換は、ユーザには見えない。バイナリ変換はしばしば、プログラムをうまく変換するためにアプリケーションプログラムコードを介する複数の通過を必要とする。その間、ソフトウェアエミュレーションを使用してアプリケーションを実行するであろう（上述した多くの欠点を伴う。）しばしば、完全な変換は達成されず、このためソフトウェアエミュレーションがなおも必要とされる。

#### 【0013】

上記の方式のいくつかの組合せが、コンピュータシステム会社およびオペレーティングシステム会社によって採用されてきた。たとえば、デジタル・イクイップメント社は、FX!32システムを提供し、マイクロソフト社（Microsoft）は、ウィンドウズNTに対するWx86拡張命令（Wx86 extension）を提供する。しかしながら、これらの方式は機能を提供するが、外部アプリケーションに所望される高性能は一般的には、満足されるものではない。

#### 【0014】

##### 【発明の開示】

以上に概略した課題は、この発明に従うホストプロセッサおよびエミュレーションコプロセッサを採用するコンピュータシステムによって大部分解決される。ホストプロセッサは、ホスト命令セットアーキテクチャによって定義される命令を実行するよう構成されるハードウェアを含み、エミュレーションコプロセッサは、ホスト命令セットアーキテクチャと異なる命令セットアーキテクチャ（「外部命令セットアーキテクチャ」）によって定義される命令を実行するよう構成されるハードウェアを含む。ホストプロセッサは、ホスト命令セットアーキテクチャ内にコード化されるアプリケーションプログラムおよびオペレーティングシステムコードを実行する。外部アプリケーションプログラムが起動されると、ホストプロセッサは、エミュレーションコプロセッサと交信して、エミュレーションコプロセッサコアに、外部アプリケーションプログラムを実行させるようにする。

#### 【0015】

有利には、外部命令セットアーキテクチャに従ってコード化されたアプリケーションプログラムは、ハードウェア内で直接に実行可能である。アプリケーションプログラムの実行性能は、ソフトウェアエミュレーションまたはバイナリ変換方法のものよりも実質的により優れているであろう。さらに、実行性能は、外部命令セットアーキテクチャを採用するプロセッサに基づくコンピュータシステム内のアプリケーションプログラムの実行性能と実質的に同様であり得、したがって外部アプリケーションプログラムのリアルタイム挙動の多くを保全するであろう。ソフトウェアエミュレーション／バイナリ変換方法およびこれらの組合せは、外部アプリケーションプログラムのハードウェア実行のために、排除され得る。エミュレーションコプロセッサは、外部命令セットアーキテクチャを実行するためのハードウェア機能を含むので、正確なアーキテクチャモデリングという困難が排除され得る。これらのさまざまな利点の組合せは、高レベル性能を提供し、外部アプリケーション実行性能がユーザに広く受け入れられることを可能にするであろう。したがって、ホスト命令セットアーキテクチャに基づくコンピュータシステムの市場支持が、増大するであろう。市場支持が増大すると、ホスト命令セットアーキテクチャのためにコード化されたアプリケーションプログラムの

数もまた増大するであろう。したがって、ホスト命令セットアーキテクチャの長期の成功および存続可能性がより高くなるであろう。

#### 【0016】

コンピュータシステム内に外部命令セットアーキテクチャのためのハードウェア機能を設けることは、さらなる利点を生み出す。特に、このコンピュータシステムは、異種のマルチプロセッシングを特徴とするであろう。エミュレーションコプロセッサが外部アプリケーションプログラムを実行している一方で、ホストプロセッサは、外部アプリケーションプログラムに関連しないオペレーティングシステムルーチンを実行するかまたは、ホストアプリケーションプログラムを実行することが可能である。有利には、このコンピュータシステムは、ホストプロセッサのみおよび、外部命令セットアーキテクチャのためのソフトウェアエミュレーション／バイナリ変換を採用するコンピュータシステムによって達成可能であるよりも、ホストおよび外部コードの両方について実質的により高いスループットを達成するであろう。

#### 【0017】

広くには、この発明は、第1のプロセッサおよび第2のプロセッサを含むコンピュータシステムのための装置を企図する。第1のプロセッサは、第1の命令セットアーキテクチャによって定義される第1の命令を実行するよう構成される。コンピュータシステムによって採用されるオペレーティングシステムは、第1の命令を用いてコード化される。第1のプロセッサに結合される、第2のプロセッサは、第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャによって定義される第2の命令を実行するよう構成される。オペレーティングシステム内で実行するよう設計されるアプリケーションプログラムは、第2の命令を用いてコード化される。第2のプロセッサは、アプリケーションプログラムを実行するよう構成されるが、その一方で第1のプロセッサはオペレーティングシステムを実行するよう構成される。加えて、第2のプロセッサは、アプリケーションプログラムのためのオペレーティングシステムルーチンの使用を検出すると、第1のプロセッサと交信するよう構成される。

#### 【0018】

この発明はさらに、第1のプロセッサと、第2のプロセッサと、オペレーティングシステムと、アプリケーションプログラムとを含む異種のマルチプロセッシングシステムを企図する。第1のプロセッサは、第1の命令セットアーキテクチャによって定義される第1の命令を実行するよう構成される。第2のプロセッサは、第1のプロセッサに結合され、第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャによって定義される第2の命令を実行するよう構成される。オペレーティングシステムは、第1の命令を用いてコード化されるが、アプリケーションプログラムは、第2の命令を用いてコード化され、オペレーティングシステム内で実行するよう設計される。第2のプロセッサは、アプリケーションプログラムを実行するよう構成されるが、第1のプロセッサは、アプリケーションプログラムに関連しないプロセスを同時に実行するよう構成される。

**【0019】**

さらに、この発明は、第1の命令セットアーキテクチャからの命令を用いてコード化され、かつ、第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャからの命令を用いてコード化されるオペレーティングシステム内で実行するよう設計されるアプリケーションプログラムを実行するための方法を企図する。アプリケーションプログラムの起動は、第2の命令セットアーキテクチャからの命令を実行するよう構成される第1のプロセッサ上で実行するオペレーティングシステムによって検出される。アプリケーションプログラムのためのコンテキストは、第1の命令セットアーキテクチャからの命令を実行するよう構成される第2のプロセッサ内で確立される。アプリケーションプログラムは、第2のプロセッサ上で実行される。

**【0020】**

この発明の他の目的および利点は、添付の図面を参照し以下の詳細な説明を読むと明らかになるであろう。

**【0021】**

この発明は、さまざまな変形および代替の態様が可能であるが、その特定の実施例が、図面の例によって示され、詳細にここに記載される。しかしながら、図面およびその詳細な説明は、この発明を、開示される特定の態様に限定するもの

ではなく、反対に、その意図は、前掲の特許請求の範囲によって定義されるようなこの発明の精神および範囲内にあるすべての変形、等価および代替を含むものである。

#### 【0022】

##### 【発明を実施するモード】

図1を参照すると、バスブリッジ12を介してさまざまなシステム構成要素に結合されるプロセッサ10を含むコンピュータシステム5のある実施例のブロック図が示される。他の実施例が可能であり企図される。例示のシステムでは、主メモリ14が、メモリバス16を介してバスブリッジ12に結合され、グラフィックスコントローラ18が、AGPバス20を介してバスブリッジ12に結合される。最後に、複数のPCIデバイス22A～22Bが、PCIバス24を介してバスブリッジ12に結合される。2次バスブリッジ26をさらに設けて、EISA/IISAバス30を介する1つ以上のEISAまたはISAデバイスへの電氣的インターフェイスを可能としてもよい。プロセッサ10は、CPUバス34を介してバスブリッジ12に結合される。

#### 【0023】

一般的に言って、プロセッサ10は、ホストプロセッサコアと、エミュレーションコプロセッサコアとを含む。ホストプロセッサコアは、ホスト命令セットアーキテクチャによって定義される命令を実行するよう構成されるハードウェアを含み、エミュレーションコプロセッサコアは、ホスト命令セットアーキテクチャと異なる命令セットアーキテクチャ（「外部命令セットアーキテクチャ」）によって定義される命令を実行するよう構成されるハードウェアを含む。ホストプロセッサコアは、オペレーティングシステムコードと、ホスト命令セットアーキテクチャ内にコード化されるアプリケーションプログラムとを実行する。外部アプリケーションプログラムが起動されると、ホストプロセッサコアは、エミュレーションコプロセッサコアと交信して、エミュレーションコプロセッサコアに外部アプリケーションプログラムを実行させる。

#### 【0024】

有利には、外部命令セットアーキテクチャに従ってコード化されたアプリケー

ションプログラムは、プロセッサ10を介してハードウェア内で直接に実行可能である。アプリケーションプログラムの実行性能は、ソフトウェアエミュレーションまたはバイナリ変換方法のそれよりも実質的に優れているであろう。さらに、実行性能は、外部命令セットアーキテクチャを採用するプロセッサに基づくコンピュータシステム内のアプリケーションプログラムの実行性能と実質的に同様であろう。ソフトウェアエミュレーション／バイナリ変換方法およびそれらの組合せは、外部アプリケーションプログラムのハードウェア実行のために、排除され得る。プロセッサ10は、外部命令セットアーキテクチャを実行するためのハードウェア機能を含むので、正確なアーキテクチャモデリングという困難が排除され得る。さらに、外部アプリケーションプログラムは、ネイティブコンピュータシステム内での実行と同様の時間期間内に実行するので、外部アプリケーションプログラムのリアルタイム挙動の多くが保全され得る。これらのさまざまな利点の組合せは、高性能を提供し、外部アプリケーション実行性能がユーザに広く受け入れられることを可能とするであろう。したがって、ホスト命令セットアーキテクチャに基づくコンピュータシステムの市場支持が増大するであろう。市場支持が増大すると、ホスト命令セットアーキテクチャのためにコード化されたアプリケーションプログラムの数も増大するであろう。したがって、ホスト命令セットアーキテクチャの長期的成功および存続可能性はより高くなるであろう。

#### 【0025】

コンピュータシステム5内に外部命令セットアーキテクチャのためのハードウェア機能を与えることは、さらなる利点を生み出す。特にコンピュータシステム5は、異種のマルチプロセッシングシステムを特徴とするであろう。エミュレーションコプロセッサが外部アプリケーションプログラムを実行している一方で、ホストプロセッサは、外部アプリケーションプログラムに関連しないオペレーティングシステムルーチンを実行可能であり、またはホストアプリケーションプログラムを実行可能である。有利には、コンピュータシステム5は、ホストプロセッサのみ、および外部命令セットアーキテクチャのためのソフトウェアエミュレーション／バイナリ変換を採用するコンピュータシステムによって達成可能であるよりも、ホストおよび外部コードの両方について実質的により高いスループッ

トを達成することが可能である。

#### 【0026】

ある特定の実施例では、ホスト命令セットアーキテクチャは、デジタル・イクイップメント社によって開発されたアルファ命令セットアーキテクチャであり、外部命令セットアーキテクチャは、x86命令セットアーキテクチャである。しかしながら、いかなる命令セットアーキテクチャが、ホスト命令セットアーキテクチャとして選択されることも可能である。たとえば、ホスト命令セットアーキテクチャは、パワーPCアーキテクチャ、インテル (Intel) によって開発されたIA-64アーキテクチャ、MIPSアーキテクチャ、SPARCアーキテクチャなどであってもよい。同様に、外部命令セットアーキテクチャは、上に列挙した例のいずれをも含むホスト命令セットアーキテクチャ以外のいかなる命令セットアーキテクチャとして選択されてもよい。

#### 【0027】

なお、コンピュータシステム5およびプロセッサ10のいくつかの異なった実施例がここに示される。図1および図2に示す実施例は、ここでの好ましい実施例として考慮されるが、ここに示す実施例のいずれも、コスト、開発計画、複雑性などを含むさまざまな設計要因に従って、好適であり得る。さらなる実施例が、前掲の特許請求の範囲の精神および範囲内で企図される。

#### 【0028】

オプションのL2キャッシュ38に結合されるプロセッサ10が、図1に示される。L2キャッシュ38は、このキャッシュがCPUバス34とは別個の専用のインターフェイスを介してプロセッサ10に結合されるので、「バックサイドL2」と呼ばれる。L2キャッシュ38は、プロセッサ10内で採用されるいかなる内部キャッシュよりも大きく、主メモリ14から達成可能であるよりもより高速のアクセスのためにデータを記憶するよう用いられ得る。

#### 【0029】

ここで用いられる、「プロセッサ」という言葉は、少なくとも、特定の命令セットアーキテクチャによって定義される命令を実行するためのハードウェアのことを言う。したがって、以下に図2に示すプロセッサコアは、この定義の下での



プロセッサであるとみなされる。プロセッサは、所望のごとくさらなるハードウェアを含んでもよい。

#### 【0030】

バスブリッジ12は、プロセッサ10と、主メモリ14と、グラフィックスコントローラ18と、P C Iバス24につながるデバイスとの間のインタフェースを与える。操作がバスブリッジ12に接続されるデバイスの1つから受取られると、バスブリッジ12は、操作のターゲット（たとえば、特定のデバイスまたは、P C Iバス24の場合には、ターゲットはP C Iバス24上にある）を特定する。バスブリッジ12は、ターゲットされたデバイスに操作を送る。バスブリッジ12は一般的には、ソースデバイスまたはバスによって使用されるプロトコルからターゲットデバイスまたはバスによって使用されるプロトコルに操作を変換する。ある実施例では、C P Uバス34は、デジタル・イクイップメント社によって開発されたE V 6バスを含み、バスブリッジ12は、アルファ（Alpha）21171または21172コア論理チップセットを含む。しかしながら、いかなるC P Uバスおよび好適なバスブリッジが使用されてもよい。

#### 【0031】

P C Iバス24に、I S A／E I S Aバスへのインタフェースを与えることに加えて、2次バスブリッジ26はさらに、所望のごとくさらなる機能を組込んでもよい。たとえば、ある実施例では、2次バスブリッジ26は、P C Iバス24の所有権を調停するためのマスタP C Iアービタ（図示せず）を含む。2次バスブリッジ26の外部またはこれと統合される、入力／出力コントローラ（図示せず）をもコンピュータシステム5内に含めて、所望のごとく、キーボードおよびマウス32ならびにさまざまな直列ポートおよび並列ポートのための動作上のサポートを与えてもよい。他の実施例では、外部キャッシュユニット（図示せず）が、プロセッサ10とバスブリッジ12との間にC P Uバス34にさらに結合されてもよい。代替的に、外部キャッシュは、バスブリッジ12に結合されてもよく、外部キャッシュのためのキャッシュ制御論理は、バスブリッジ122統合されてもよい。

#### 【0032】

主メモリ14は、アプリケーションプログラムが記憶され、プロセッサ10が主に実行するメモリである。好適な主メモリ14は、DRAM (Dynamic Random Access Memory) および、好ましくはSDRAM (Synchronous DRAM) の複数個のバンクを含む。

#### 【0033】

PCIデバイス22A~22Bは、たとえば、ネットワークインターフェイスカード、ビデオアクセラレータ、オーディオカード、ハードもしくはフロッピーディスクドライブまたはドライブコントローラ、SCSI (スモールコンピュータシステムインターフェイス) アダプタおよび電話機能カードなどの、さまざまな周辺デバイスを例示するものである。同様に、ISAデバイス28は、モデム、サウンドカードおよび、GPIBまたはフィールドバスインターフェイスカードなどのさまざまなデータ収集カードなどの、さまざまなタイプの周辺デバイスを例示するものである。

#### 【0034】

グラフィックスコントローラ18は、ディスプレイ36上のテキストおよび画像のレンダリングを制御するために設けられる。グラフィックスコントローラ18は、先行技術に一般的に公知の典型的なグラフィックスアクセラレータを採用して、主メモリ14におよびそこから効果的にシフト可能である3次元のデータ構造をレンダリングすることが可能である。したがって、グラフィックスコントローラ18は、バスブリッジ12内のターゲットインターフェイスへのアクセスを要求しかつ受取ることによって主メモリ14へのアクセスを獲得することが可能であるという点で、AGPバス20のマスタであり得る。専用のグラフィックスバスが、主メモリ14からのデータの高速の取出しを可能とする。ある操作では、グラフィックスコントローラ18は、AGPバス20上のPCIプロトコルトランザクションを生成するようさらに構成されてもよい。したがって、バスブリッジ12のAGPインターフェイスは、AGPプロトコルトランザクションと、PCIプロトコルターゲットおよびイニシエータトランザクションとの両方をサポートする機能を含み得る。ディスプレイ36は、画像またはテキストが与えられることが可能である任意の電子ディスプレイである。好適なディスプレイ3

6は、陰極線管（「CRT」）、液晶ディスプレイ（「LCD」）などを含む。

#### 【0035】

なお、AGPバス、PCIバスおよびISAバスまたはEISAバスが、上記の記載では例として用いられたが、いかなるバスアーキテクチャが所望のごとく置換えられてもよい。さらになお、コンピュータシステム5は、付加的プロセッサ（たとえば、オプションのL2キャッシュ38aとともに、コンピュータシステム5のオプションの構成要素として示されるプロセッサ10a）を含むマルチプロセッシングコンピュータシステムであつてもよい。プロセッサ10aは、プロセッサ10と同様であつてもよい。より特定のには、プロセッサ10aは、プロセッサ10の同一のコピーであつてもよい。図1に示すとおり、プロセッサ10aは、CPUバス34と同様の別個のCPUバス34aを介してバスブリッジ12に結合される。代替的に、プロセッサ10aは、プロセッサ10とCPUバス34を共有してもよい。

#### 【0036】

図2を参照すると、プロセッサ10の第1の実施例のブロック図が示される。他の実施例が可能であり企図される。図2の実施例では、プロセッサ10は、バスインターフェイスユニット40と、メモリ管理ユニット（MMU）42と、命令キャッシュ（Iキャッシュ）44と、データキャッシュ（Dキャッシュ）46と、ホストプロセッサコア48と、エミュレーションコプロセッサコア50とを含む。バスインターフェイスユニット40は、CPUバス34に、およびL2キャッシュ38へのバックサイドL2インタフェース52に結合される。バスインタフェースユニット40はまた、MMU42に結合され、これはさらに、命令キャッシュ44に、およびデータキャッシュ46に結合される。命令キャッシュ44およびデータキャッシュ46はどちらも、ホストプロセッサコア48に結合され、データキャッシュ46は、エミュレーションコプロセッサコア50に結合される。命令キャッシュ44は、さらに以下に詳細に記載するとおり、エミュレーションコプロセッサ50にオプションで結合される。図2の実施例では、プロセッサ10の素子は、半導体基板上に集積される。コマンドインターフェイス54は、ホストプロセッサコア48とエミュレーションコプロセッサ50との間に結

合される。

#### 【0037】

ホストプロセッサコア48は、命令キャッシュ44からの命令をフェッチしかつ、これらの命令を実行するよう構成される。命令は、ホストアプリケーションプログラムの一部を含んでもよく、またはコンピュータシステム5によって採用されるオペレーティングシステムの一部を含んでもよい。オペレーティングシステムのある特定の部分を用いて、外部アプリケーションプログラムの起動を含む、プロセスを生成する。オペレーティングシステムのプロセス生成部分の実行中に、外部アプリケーションプログラムが起動されていると検出されれば、ホストプロセッサコア48は、コマンドインターフェイス54を介してエミュレーションコプロセッサコア50と交信する。ホストプロセッサコア48は、起動されている外部アプリケーションプログラムに対応するエミュレーションコプロセッサコア50内のコンテキストを確立する。コンテキスト内に含まれるのは、初期プログラムカウンタアドレスであり、外部アプリケーションプログラム内の第1の命令は、ここからフェッチされる。コンテキストが一旦確立されると、ホストプロセッサコア48は、エミュレーションコプロセッサコア50にコマンドを与えて、命令を開始する。エミュレーションコプロセッサコア50は、プログラムカウンタアドレスの命令のフェッチを開始し、外部命令セットアーキテクチャに従って命令を実行する。ここで用いられる「コンテキスト」という言葉は、プロセスに特定の値を言う。コンテキストは、一般的には、プロセスに割当てられるメモリページと、レジスタ値とを含む。

#### 【0038】

エミュレーションコプロセッサコア50は、外部アプリケーションプログラムプロセス内で、ホスト命令セットアーキテクチャ内にコード化された命令への遷移が生じているかどうかを決定するよう構成される。たとえば、外部アプリケーションプログラムがオペレーティングシステムルーチンをコールした場合、オペレーティングシステムはホスト命令セットアーキテクチャに従ってコード化されているので、遷移が検出される。加えて、オペレーティングシステムコードまたは、ホスト命令セットアーキテクチャを用いる他のコードを導く例外および他の

プロセッサ事象は、遷移である。遷移が生じていると決定すると、エミュレーションコプロセッサコア50は、コマンドインターフェイス54を介してホストプロセッサコア48と、エミュレーションコプロセッサコア50が停止したことを交信する。ホストプロセッサコア48は、停止のための理由を決定するためにコンテキスト情報を要求し、対応する動作をとる（たとえば、コールされたルーチンを実行するかまたは、オペレーティングシステムサービスを与える）。一旦、ホストプロセッサコア48が、外部アプリケーションプログラムが再開可能であると決定すると、ホストプロセッサコア48は、（もし必要であれば）コンテキスト情報を与え、エミュレーションコプロセッサコア50が開始するためのコマンドを与える。

#### 【0039】

コマンドインターフェイス54は、さまざまな態様で実現可能である。たとえば、コマンドインターフェイス54は、ホストプロセッサコア48とエミュレーションコプロセッサコア50との間の1組のハードワイヤード信号を含んでもよい。コマンド信号は、コマンドインターフェイス54のために定義される各コマンドに割当てられてもよく、さらに、コンテキスト値を送るためのバスに割当てられてもよい。代替的に、コマンドインターフェイス54は、プロセッサコア間の通信のためのFIFO（すなわち、ホストプロセッサコア48からエミュレーションコプロセッサコア50へのメッセージのための1つ以上のFIFOおよび、エミュレーションコプロセッサコア50からホストプロセッサコア48へのメッセージのための1つ以上のFIFO）を含んでもよい。なお、コマンドインターフェイス54は、「通信チャネル」の一例であり得る。一般的に、通信チャネルは、伝送器と受信器との間の接続であり、これを介してメッセージが送信されることが可能である。予め定義されたプロトコルを用いて、チャネルを介して伝送されるメッセージを定義してもよい。たとえば、ハードワイヤード信号は、通信チャネルを形成し、信号の組合せは、メッセージを伝送するために用いられる。さらに、FIFOは、通信チャネルを形成してもよく、メッセージはFIFOエントリとして符号化される。FIFOは単に、メモリ内のキューとして維持されてもよい。

## 【0040】

ホストプロセッサコア48およびエミュレーションコプロセッサコア50は、この実施例では、命令キャッシュ44およびデータキャッシュ46を共有する。ホストプロセッサコア48は、命令キャッシュ44から命令をフェッチし、データキャッシュ40からの命令に応答して操作されるべきデータをフェッチする。エミュレーションコプロセッサコア50はまた、データキャッシュ46からデータをフェッチする。いくつかの実施例が、エミュレーションコプロセッサコア50のための命令ソースについて企図される。第1の実施例では、エミュレーションコプロセッサコア50は、命令キャッシュ44から命令をフェッチし、第2の実施例では、エミュレーションコプロセッサコア50は、データキャッシュ46から命令をフェッチする。いくつかの要因が、エミュレーションコプロセッサコア50が、命令キャッシュ44からまたはデータキャッシュ46から命令をフェッチするののかについての決定に影響を及ぼし得る。たとえば、エミュレーションコプロセッサコア50がx86命令セットアーキテクチャを実行するような実施例では、自己変更コードなどの特徴がサポートされる。したがって、命令キャッシュ44は、データキャッシュ46に対する更新を詮索して、そのような状況を検出し得る。しかしながら、ホスト命令セットアーキテクチャは、そのような特徴をサポートしないかも知れず、命令キャッシュ44によるデータキャッシュ46の詮索は、不必要かもしれない。さらに、ホストプロセッサコア48は、データとしてエミュレーションコプロセッサコア50によって実行されるべき命令にアクセスし得る。たとえば、エミュレーションコプロセッサコア50によって実行される外部アプリケーションプログラムについての例外サービスを与えるために、ホストプロセッサコア48は、例外が生じた命令を調べる必要がないかもしれない。したがって、エミュレーションコプロセッサコア50についての命令は、データキャッシュ46内に既に記憶されている可能性がある。さらに別の企図された実施例では、エミュレーションコプロセッサコア50は、命令キャッシュを含み、命令キャッシュミスは、データキャッシュ46からフェッチされる。

## 【0041】

ホストプロセッサコア48およびエミュレーションコプロセッサコア50は、

この実施例でも同様に、MMU 42を共有する。MMU 42は、ホストプロセッサコア48およびエミュレーションコプロセッサコア50内の命令の実行によって生成される仮想アドレスから、主メモリ14またはL2キャッシュ38を讀出するためにバスインターフェイスユニット40が使用するであろう物理的アドレスに変換するよう構成される。命令キャッシュ44およびデータキャッシュ46はまた、物理的アドレスにしたがって命令およびデータを記憶してもよく、この場合には、MMU 42は、命令キャッシュ44およびデータキャッシュ46に並列にアクセス可能である。

#### 【0042】

一般的には、ホスト命令セットアーキテクチャおよび外部命令セットアーキテクチャは、異なったアドレス変換メカニズムを定義する。MMU 42は、ホスト命令セットアーキテクチャによって定義されるアドレス変換メカニズムをサポートし得り、ホストプロセッサコア48およびエミュレーションコプロセッサコア50の両方についての変換は、ホストアドレス変換メカニズムから与えられ得る。異なったページサイズが、ホストおよび外部命令セットアーキテクチャについて定義された場合、所望であれば、変換メカニズムの保護部分は、保護情報のさらなるコピーで増強され、より小さいページサイズの粒度に対して独立した保護を与えることが可能である。代替的に、MMU 42は、ホスト命令セットアーキテクチャによって定義されるアドレス変換メカニズムと、外部命令セットアーキテクチャによって定義されるアドレス変換メカニズムとをサポートするよう構成されてもよい。オペレーティングシステムは、ホスト命令セットアーキテクチャによって定義されるアドレス変換メカニズムにしたがって、メモリのページを仮想アドレスのために割当ててもよい。さらなるソフトウェアまたはMMU 42内のハードウェアが、外部命令セットアーキテクチャによって定義されるアドレス変換メカニズムを用いて対応する変換を生成してもよい。代替的に、オペレーティングシステムは、ページが外部アプリケーションプログラムによって要求された場合、外部命令セットアーキテクチャのアドレス変換メカニズム内のアドレス変換を生成してもよい。

#### 【0043】

図2に示すとおり、ホストプロセッサコア48は、フェッチ／デコードユニット60と、複数の機能ユニット62A～62Cと、オーダおよび依存性制御ブロック64と、複数のレジスタ66とを含む。同様に、エミュレーションコプロセッサコア50は、フェッチ／デコードユニット70と、複数の機能ユニット72A～72Cと、オーダおよび依存性制御ブロック74と、複数のレジスタ76とを含むものとして示される。一般的には、フェッチ／デコードユニット60および70は、対応する命令セットアーキテクチャによって定義される命令をフェッチし、これらの命令をデコードしてどの対応する機能ユニット62A～62Cおよび72A～72Cが命令を実行するよう構成されるかを決定するよう構成される。フェッチ／デコードユニット60および70は、機能ユニット62A～62Cおよび72A～72Cと、オーダおよび依存性制御ブロック64および74とにそれぞれ、命令を与えることが可能である。オーダおよび依存性制御ブロック64および74は、命令依存性が検出され、オペランド値についての適切なソースが各命令ごとに与えられることを確実にするだけでなく、命令実行オーダが適切に維持されることを保証する。オーダおよび依存性制御ブロック64および74は、たとえば、リオーダバッファおよび関連する回路構成を含んでもよい。代替的に、オーダおよび依存性制御ブロック64および74は、オーダ制御機能および依存性制御機能を実行するためのいかなる好適な回路構成を含んでもよい。さらに別の代替例では、オーダ操作および依存性操作は、フェッチ／デコードユニット60および70によって実行されてもよい。レジスタ66および76は、対応する命令セットアーキテクチャによって定義されるレジスタである。

#### 【0044】

機能ユニット62Aおよび72Aは、図2の実施例ではデータキャッシュ46に接続されるものとして示される。これらの機能ユニットは、メモリ操作（すなわち、ロードおよび記憶）機能を含んでもよい。代替の実施例では、他の機能ユニットが同様にメモリ操作機能を含んでもよい。機能ユニット62A～62Cの組合せは、ホスト命令セットアーキテクチャによって定義される命令を実行するために用いられるハードウェアを与える。同様に、機能ユニット72A～72C



の組合せは、外部命令セットによって定義される命令を実行するために用いられるハードウェアを与える。所望であれば、マイクロコード技術を採用して、機能ユニット設計を簡素にしてもよい。なお、複数の機能ユニットが、図2のコア48および50の各々に示されるが、コア48および50の1つまたはその両方に1つの機能ユニットを有する実施例を含む、より多くのまたはより少ない機能ユニットを有する実施例が企図される。さらに、コア48または50のいずれかが他方よりも多い機能ユニットを有してもよい。

#### 【0045】

なお、1個のエミュレーションコプロセッサコアが図2に示されるが（1個のエミュレーションコプロセッサが以下の図7、9、10、11および13に示される）、複数のエミュレーションコプロセッサが採用されてもよいことが企図される。さらに、複数の外部命令セットアーキテクチャが、複数のエミュレーションコプロセッサを用いてサポートされてもよいことが企図される。

#### 【0046】

図3を参照すると、コンピュータシステム5のある実施例によって採用されるソフトウェアモデルのブロック図が示される。図3は、外部アプリケーションプログラム82を含むホストプロセス80を例示する。図示の実施例は、たとえば、ホスト命令セットアーキテクチャとしてのアルファ命令セットアーキテクチャと、外部命令セットアーキテクチャとしてのx86命令セットアーキテクチャとともに、ウィンドウズNTオペレーティングシステムの動作を表わす。図3は、他のオペレーティングシステム、ホスト命令セットアーキテクチャおよび外部命令セットアーキテクチャをさらに表わしてもよい。他の実施例が可能であり企図される。

#### 【0047】

外部アプリケーション82は、外部命令セットアーキテクチャ内にコード化される1つ以上のモジュールを含む。外部アプリケーションは、オペレーティングシステムルーチンへのコールを含んでもよい。オペレーティングシステムルーチンを直接コールする代わりに、各ルーチンは、「サンク(thunk)」と置換えられる。サンクは、これが置換わるルーチンと同じ名前（および、したがって、プ

ロセス80のアドレス空間内の同じアドレス)を有するルーチンである。この実施例では、サンクは、特定の予め定義された違法操作コードを含み、これによってエミュレーションコプロセッサは違法操作コードトラップ(または「例外」)をとる。違法操作コードトラップをとると、エミュレーションコプロセッサは、ホストプロセッサと交信して、外部アプリケーションが停止したことを示す。たとえば、エミュレーションコプロセッサは、違法操作コードトラップをとるとストップメッセージを生成するハードウェアを含んでもよい。代替的に、違法操作コードトラップハンドラ(違法操作コードトラップが発生するとフェッチされるよう定義される予め定められたアドレスに記憶されるコード)が、ストップメッセージを与えるためにコード化されてもよい。サンクの2つの組、すなわちオペレーティングシステムサンク86およびプロセスサンク88が図3に示される。オペレーティングシステムサンク86は、外部アプリケーションプログラム82にコード化される直接オペレーティングシステムコールと、外部アプリケーションプログラム82の実行中の例外に対する応答として生じる間接オペレーティングシステムコールとの両方の、オペレーティングシステムコールを捕えるために用いられる。加えて、プロセスサンク88は、プロセス内に含まれるホストコード90のブロックと交信するために含まれてもよい。しかしながら、プロセスサンク88およびホストコード90は、オプションである。上述したプロセスは、外部アプリケーションコードとホストコードとの間の遷移を検出するために用いることが可能である。他の実施例が、遷移を検出するための他の方法を採用してもよい。

#### 【0048】

ホストプロセス80は、ホストプロセッサとエミュレーションコプロセッサとの間で交信するために用いられ得るエミュレーションインターフェイスコード92をさらに含む。したがって、オペレーティングシステムサンク86は、エミュレーションインターフェイスコード92の呼出につながり、ホストプロセッサとエミュレーションコプロセッサとの間でメッセージをやり取りし得る。さらに、ホストプロセッサは、エミュレーションインターフェイスコード92を用いてエミュレーションコプロセッサからコンテキスト情報を要求するよう構成されても

よい。外部アプリケーションプログラム82によってコールされるオペレーティングシステムルーチンと、オペレーティングシステム84によって与えられる対応するオペレーティングシステムルーチンとは同じ機能を与えるが、命令セットアーキテクチャが異なるために、コール協定（すなわち、アプリケーションとオペレーティングシステムルーチンとの間でパラメータがやり取りされる態様）は異なっている。たとえば、レジスタの数およびタイプが異なっていると、レジスタ内のパラメータを送る能力（メモリ場所とは対照的に）は異なる。したがって、エミュレーションインターフェイスコード92は、コールについてのパラメータであるコンテキスト値を要求可能であり、パラメータをホストプロセッサ上の対応するレジスタ内に与えることが可能である。オペレーティングシステムコールは次に、ホストプロセッサによって実行され得る。その後に、オペレーティングシステムルーチンの結果は、コール協定の変換を逆転させることによってエミュレーションコプロセッサに与えられ得る。

#### 【0049】

さらになお、オペレーティングシステムライブラリコード94が、ホストプロセッサ80内に含まれ得る。たとえば、ウィンドウズNTオペレーティングシステム内に定義されるダイナミックロードライブラリが、オペレーティングシステムライブラリ94を介して解決可能である。

#### 【0050】

図4を参照すると、図1に示すコンピュータシステムのある実施例に従うアプリケーションプログラムの初期化を例示するフローチャートが示される。他の実施例が可能であり企図される。いくつかのステップが、わかりやすくするために直列順序で図4に示され得るが、いかなる好適な順序が用いられてもよい。さらに、所望のごとくステップが並列に実行されてもよい。

#### 【0051】

アプリケーションプログラムを起動するためにユーザからコマンドを受取ると、オペレーティングシステムは、アプリケーションプログラムが実行するプロセスを生成する。オペレーティングシステムは、アプリケーションプログラムのファイルフォーマットを調べ、どのタイプのコードがアプリケーションプログラム

内に含まれるかを決定する（ステップ100）。たとえば、ウィンドウズNTオペレーティングシステムを採用する実施例では、ポータブル実行フォーマットは、どの命令セットアーキテクチャにアプリケーションプログラムがコード化されるかについての表示を含む。ポータブル実行フォーマットは、ウィンドウズNTによって定義されるアプリケーションプログラミングインターフェイスの一部として定義される。

#### 【0052】

アプリケーションプログラムが、ホスト命令セットアーキテクチャに従ってコード化されるべきであると決定されると（決定ブロック102）、オペレーティングシステムは、通常のホストプロセスとしてプロセスを確立し、アプリケーションプログラムがホストプロセッサによって実行される（ステップ104）。他方で、アプリケーションプログラムが、ホスト命令セットアーキテクチャに従ってコード化されるべきでないと決定された場合、オペレーティングシステムは、アプリケーションプログラムがコンピュータシステム内のエミュレーションコプロセッサによって実行可能である外部命令セットアーキテクチャにしたがってコード化されるかどうかを決定する（決定ブロック106）。外部命令セットアーキテクチャがエミュレーションコプロセッサによって実行可能である場合、オペレーティングシステムは、エミュレーションコプロセッサ上の外部アプリケーションプログラムを起動するためにエミュレーションコプロセッサインターフェイスコードを呼出す（ステップ108）。外部命令セットアーキテクチャがエミュレーションコプロセッサによって実行可能でない場合、オペレーティングシステムは、メッセージをユーザに表示して、アプリケーションがサポートされていないことを示す（ステップ110）。アプリケーションプログラムは、この場合起動されない。代替的に、アプリケーションのソフトウェアエミュレーションまたはバイナリ変換が、所望であれば、ステップ110で設けられてもよい。たとえば、デジタル・イクイップメント社のFX!32製品またはマイクロソフトのWinx86製品と同様の方式が採用されてもよい。

#### 【0053】

図5を参照すると、図3に示すエミュレーションインターフェイスの呼出（た

例えば、図4に示すステップ108)のある実施例を例示するフローチャートが示される。他の実施例が可能であり企図される。プロセスコンテキストは、(ホストプロセッサとエミュレーションコプロセッサとの間のコマンドインターフェイスを介して伝送されるコマンドを用いて)ホストプロセッサによって確立される。アプリケーションプログラム内の第1の命令の仮想アドレスであるプログラムカウンタレジスタについての値を含む、レジスタについての初期値が与えられる。コンテキストを確立した後、「ゴー」(すなわち実行の開始)コマンドが、エミュレーションコプロセッサに与えられる(ステップ120)。

#### 【0054】

ホストプロセッサ上で実行するエミュレーションインターフェイスコードは、コマンドインターフェイス54をモニタして、ホストコードへの遷移が検出されたことを示すエミュレーションコプロセッサからのメッセージを受取る(すなわち、ストップメッセージがエミュレーションコプロセッサから受取られる)。ホストコードへの遷移が検出されると(決定ブロック122)、ホストプロセッサは、遷移がプロセス脱出条件によるものかどうかを決定する(決定ブロック128)。図6に以下に例示するように、ストップコマンドは、停止のための理由についての表示を含み得る。プロセス脱出が検出された場合、破壊プロセスメッセージがオペレーティングシステムに送信され、エミュレーションインターフェイスコードが存在する(ステップ130)。

#### 【0055】

他方で、プロセス脱出が検出されなかった場合、ホストプロセッサは、コマンドインターフェイス54を介してコンテキスト情報を集めて、どのオペレーティングシステムルーチンが実行されるべきかおよび、コールパラメータは何かを決定する(ステップ124)。ホストコードは次に、ホストプロセッサ上で実行される。コンテキスト情報は、コマンドインターフェイス54を介してエミュレーションコプロセッサに与えられる。オペレーティングシステムルーチンの実行によって与えられた結果は、適用可能であれば、この態様でエミュレーションコプロセッサに送られることが可能である。次に、ゴーコマンドが与えられることによって、エミュレーションコプロセッサは続行し(ステップ126)、ホストプ

ロセッサは引き続き、エミュレーションコプロセッサからのメッセージをモニターする。

#### 【0056】

なお、外部アプリケーションプログラムによってコール可能であるオペレーティングシステムルーチンには少なくとも2つのタイプがある。第1のタイプは、外部アプリケーションプログラム内に意図的にコード化されるオペレーティングシステムライブラリルーチンコールである。ライブラリルーチンは、多くのアプリケーションプログラムによって使用され得る、低レベルサービスを与えサービス自体をコード化する代わりにアプリケーションプログラムによって使用される。典型的には、ライブラリルーチンおよびルーチンによって用いられるパラメータは、アプリケーション開発者の使用のためにドキュメントされる。加えて、例外処理を与えるオペレーティングシステムルーチンがコールされ得る。名前が意味するとおり、エミュレーションコプロセッサが例外を検出したときこれらのルーチンは「コールされる」。たとえば、命令フェッチアドレスまたはデータアドレスが変換を失敗したときに生じるページフォルトが、例外ルーチンを呼出しページを割当ててゐる。

#### 【0057】

特定のページへの初期アクセスの際にページフォルトは起こり得る。たとえば、エミュレーションコプロセッサが、アプリケーションプログラムの第1の命令をフェッチしようとしたとき、第1の命令を含むページはまだ、アプリケーションプログラムに割当てられていない可能性がある。したがって、フェッチアドレスは変換せず、ページフォルトが生じる。同様に、データが新しいページからアクセスされるたびに、ページフォルトが起こり得る。ページがディスクに「ページアウト」されて異なったページの割当てが可能とされたときにも、ページフォルトは起こり得る。

#### 【0058】

なお、図5のフローチャートは、ウィンドウズNTなどのプリエンティティブ・マルチタスクオペレーティングシステム下で割込まれて、ホストプロセッサが他のタスク（たとえば、ホストアプリケーションプログラムまたは実行されてい

るアプリケーションに関連しないオペレーティングシステムルーチン) を実行することを可能にすることもある。さらに、複数の外部アプリケーションが同時に実行している場合、複数のプロセスがメッセージをモニタしている可能性がある。

#### 【0059】

ある実施例では、エミュレーションインターフェイスコードは、ウィンドウズNTオペレーティングシステムに対するWx86拡張命令にインターフェイスし得る。

#### 【0060】

図6を参照すると、コマンドインターフェイス54のある実施例によってサポートされるコマンドを例示する表140が示される。異なったコマンドまたは、異なったコマンドと表140に示す1つ以上のコマンドとの組合せを採用する他の実施例が企図される。

#### 【0061】

レジスタ読出コマンドが、ホストプロセッサによるエミュレーションコプロセッサレジスタの読出のためにサポートされる。エミュレーションコプロセッサは、要求されたレジスタ値を与えることによってレジスタ読出コマンドに応答する。なお、メモリ値もまた同様に、エミュレーションコプロセッサのコンテキストから読出可能である。しかしながら、エミュレーションコプロセッサおよびホストプロセッサは、同じ物理的メモリを共有するので、ホストプロセッサは、メモリ値を直接に読出することが可能である。上述したとおり、同じ変換がホストプロセッサおよびエミュレーションコプロセッサの両方によって共有されるか、または、変換は、外部アプリケーションプログラムに割当てられた各ページごとに、ホストプロセッサの命令セットアーキテクチャおよびエミュレーションコプロセッサの命令セットアーキテクチャの両方に従って生成される。したがって、ホストプロセッサは、外部アプリケーションのコンテキストに割当てられたメモリを見るであろう。

#### 【0062】

同様に、ホストプロセッサがエミュレーションコプロセッサ内のレジスタを更

新することを可能にするためにレジスタ書込コマンドがサポートされる。エミュレーションコプロセッサは、レジスタ書込コマンド内に与えられたデータを受取り、受取った値で特定のレジスタを更新する。メモリの読出に関する前の記載と同様に、ホストプロセッサは、同様に、エミュレーションコプロセッサのコンテキスト内のメモリを更新することが可能である。

#### 【0063】

ゴーコマンドは、エミュレーションコプロセッサが実行を開始すべきかどうかをエミュレーションコプロセッサに示す。エミュレーションコプロセッサにゴーコマンドを送るより前に、実行ポインタが、エミュレーションコプロセッサ内のプログラムカウンタレジスタ内に記憶される。ゴーコマンドを受取ると、エミュレーションコプロセッサは、実行ポインタの命令のフェッチおよび実行を開始する。代替的に、実行ポインタは、所望であれば、ゴーコマンド内で交信されてもよい。

#### 【0064】

外部アプリケーションプログラムの実行のためにアーキテクチャ的切換が実行されるべきである（たとえば、ホストコードが実行されるべきである）ことが決定されると、ストップコマンドがエミュレーションコプロセッサによって伝送される。ストップコマンドは、エミュレーションコプロセッサが停止したことをホストプロセッサに通知し、また停止の理由を与える。停止のための様々な理由が所望のごとく採用され得る。たとえば、停止のための理由は、（i）オペレーティングシステムコールのために（上述したような）サンクを実行すること、（ii）外部アプリケーションプログラムの実行の終了を検出すること、または（iii）アプリケーションプログラムの実行中に例外を経験することを含み得る。所望であれば、レジスタ読出コマンドを用い、外部アプリケーションプログラムのメモリを読出して、ホストプロセッサはさらなるコンテキスト情報を集めてもよい。

#### 【0065】

なお、「メッセージ」という言葉は、ホストプロセッサとエミュレーションコプロセッサとの間の交信を言うものとしてここでは用いられ得る。メッセージと



いう言葉とコマンドとはこの開示では同義であることが意図される。

【0066】

図7を次に参照すると、プロセッサ10の第2の企図された実施例が示される。図7の実施例は、たとえば、図1に示すコンピュータシステム5の実施例において採用されてもよい。図3～図6は、一般的には、図7の実施例にも適用可能である。他の実施例が可能であり企図される。図7に示すとおり、プロセッサ10は、エミュレーションコプロセッサ150と、ホストプロセッサ152と、インターフェイス論理ユニット154とを含む、エミュレーションコプロセッサ150およびホストプロセッサ154は、インターフェイス論理ユニット154に結合され、これはさらにCPUバス34に結合される。ホストプロセッサ152はさらに、バックサイドL2インターフェイス52を介してL2キャッシュ38に結合される。

【0067】

エミュレーションコプロセッサ150は、図2に示すものと同様のエミュレーションコプロセッサコア50と、命令キャッシュ44およびデータキャッシュ46と同様のキャッシュと、MMU42と同様のMMUとを含んでもよい。ホストプロセッサ152は、図2に示すものと同様のホストプロセッサコア48と、命令キャッシュ44およびデータキャッシュ46と同様のキャッシュと、MMU42と同様のMMUとを含んでもよい。

【0068】

ある特定の実施例に従うと、図7に示すようなプロセッサ10は、プリント回路基板に装着される3つの別個の半導体チップを含む。プリント回路基板は、エッジコネクタを含み、エンキャプシュレーションされてコンピュータシステム5内に含まれてもよい。たとえば、プロセッサ10は、インテルおよびアドバンスド・マイクロ・デバイシズ (Advanced Micro Devices) によって開発されたスロット1、スロットAまたはスロット2000の仕様のいずれかに従って設計されてもよい。あるチップは、エミュレーションコプロセッサ150を実現する。第2のチップは、ホストプロセッサ152を実現し、第3のチップは、インターフェイス論理154を実現する。たとえば、エミュレーションコプロセッサ15

0 およびホストプロセッサ152は、カスタムデザイン半導体チップであってもよく、インターフェイス論理ユニット154は、特定用途向け集積回路（ASIC）、フィールドプログラマブルゲートアレイ（FPGA）などであってもよい。同様に、インターフェイス論理ユニット154をカスタム半導体チップとして実現することを含む、他の構成が可能であり企図される。

#### 【0069】

図7に示す実施例は、（異なった半導体製造プロセスを用いておそらく製造される）予め設計されたエミュレーションコプロセッサ150およびホストプロセッサ152を用いてプロセッサ10を形成することを見込む。エミュレーションコプロセッサ150およびホストプロセッサ152は各々、インターフェイス論理ユニット154へのバスインターフェイス（それぞれ、参照番号156および158）を与え得る。たとえば、バスインターフェイス156および158は、CPUバス34に論理的および電氣的に同一であってもよい。代替的に、バスインターフェイス156および158は、CPUバス34に専用のものとは異なったバスプロトコルおよび／または電氣的仕様に従って動作可能である。さらになお、バスインターフェイス156は、バスインターフェイス158と異なってもよく、インターフェイス論理ユニット154は、バス上のトランザクションを、バスブリッジ12の動作と同様の適切なプロトコルに変換可能である。

#### 【0070】

インターフェイス論理ユニット154は、図7の実施例では、コマンドインターフェイス機能（たとえばコマンドインターフェイス54）を与える。メモリおよびI/Oバスサイクルと異なる予め定義されたバスサイクルが、バスインターフェイス156および158上に定義されて、エミュレーションコプロセッサ150とホストプロセッサ152との間にさまざまなコマンドを交信してもよい。代替的に、コマンドインターフェイス52が、1組のFIFOを含んで、エミュレーションコプロセッサ150およびホストプロセッサ152はこの中にコマンドを書込み、エミュレーションコプロセッサ150およびホストプロセッサ152はここからコマンドを読出してもよい。

#### 【0071】

コマンドインターフェイス機能を与えることに加えて、インターフェイス論理ユニット154は、エミュレーションコプロセッサ150およびホストプロセッサ152から、CPUバス34および任意で非要求プロセッサに、非コマンド（たとえばメモリおよびI/O）を送る。図8は、インターフェイス論理ユニット154のある実施例に従うコマンド要求および非コマンド要求の両方を送るある実施例を例示するフローチャートである。他の実施例が可能であり企図される。図8に示すステップは時に、わかりやすくするために直列順で例示される。しかしながら、ステップは、所望のごとく、いかなる好適な順序で実行されてもよく、並列に実行されてもよい。

#### 【0072】

（ホストプロセッサ152およびエミュレーションコプロセッサ154の両方に送られる、CPUバス34に対するコヒーレンシ要求以外の）要求は、ホストプロセッサ152またはエミュレーションコプロセッサ150のいずれかによって起動される。要求がバスインターフェイス156上で受取られた場合、要求はエミュレーションコプロセッサ150によって起動される。要求がホストプロセッサ158によって受取られた場合、要求はホストプロセッサ152によって起動される。インターフェイス論理ユニット154は、要求のイニシエータを決定する（決定ブロック160）。要求がホストプロセッサ152によって起動された場合、インターフェイス論理ユニット154は、要求がエミュレーションインターフェイスについてのコマンド（たとえば、コマンドインターフェイス54を介するエミュレーションコプロセッサ150へのコマンド）であるかどうかを決定する（決定ブロック162）。要求がエミュレーションコプロセッサ150へのコマンドであった場合、要求はエミュレーションコプロセッサ150に送られる（ステップ164）。CPUバス34は、コマンドによって影響を受けることはないであろう。要求がエミュレーションコプロセッサ150へのコマンドでなければ、インターフェイス論理ユニット154は、コマンドをCPUバス34に送る（ステップ166）。

#### 【0073】

他方で、要求がエミュレーションコプロセッサ150から受取られた場合、要

求はホストプロセッサ152に送られる(ステップ168)。要求のデスティネーションがホストプロセッサ152であるため、エミュレーションインターフェイスコマンドはホストプロセッサ152に送られる。メモリ要求およびI/O要求は、エミュレーションコプロセッサ150がホストプロセッサ152のL2キャッシュ資源(たとえばL2キャッシュ38)を共有することを可能にするために、ホストプロセッサ152に送られる。メモリ要求は、ホストプロセッサ152が要求されたデータを与えるように、コピーレンシ要求の形でインターフェイス論理ユニット154によって与えられてもよい。代替的に、インターフェイス論理ユニット154は、バスインターフェイス158に従って与えられるバスサイクルとは異なる予め定義されたバスサイクルを採用して、L2キャッシュ38の読出を要求してもよい。この態様では、コスト削減が、ホストプロセッサ152とエミュレーションコプロセッサ150との間で共有されるL2キャッシュを採用することによって達成可能である。

#### 【0074】

上述したとおり、エミュレーションコプロセッサ150からの要求は、エミュレーションインターフェイスについてのコマンド(たとえば、コマンドインターフェイス54または予め定められたバスサイクルを介するホストプロセッサ152へのコマンドー決定ブロック170)または、メモリもしくはI/O要求のいずれであってもよい。要求がエミュレーションインターフェイスコマンドであった場合、要求はホストプロセッサ152に送られることが可能であり(ステップ168)、さらなる動作は必要とされないであろう。他方で、要求がエミュレーションインターフェイスコマンドでなかった場合、インターフェイス論理ユニット154は、これに送られるバスサイクルに対するホストプロセッサ152の応答(ステップ168)から、要求がホストプロセッサ152によって満足され得るかどうかを決定する(決定ブロック172)。要求がホストプロセッサ152によって満足され得るのであれば、ホストプロセッサ152によって与えられたデータは、インターフェイス論理ユニット154を介してエミュレーションコプロセッサ150に送られる(ステップ174)。要求がホストプロセッサ152によって満足され得ないならば、要求は、インターフェイス論理ユニット154

によってCPUバス34に送られる（ステップ166）。

#### 【0075】

次に図9を参照すると、たとえば、図1に示すコンピュータシステム5内に採用可能であるプロセッサ10の第3の実施例のブロック図が示される。図3～図6は一般的に、この実施例にも適用可能である。他の実施例が可能であり企図される。図9の実施例では、プロセッサ10は、エミュレーションコプロセッサ150とホストプロセッサ152とを含む。より詳細に示されるホストプロセッサ152は、コア48と、Iキャッシュ44と、Dキャッシュ46と、MMU42と、バスインターフェイスユニット40とを含む。エミュレーションコプロセッサ150は、コマンドインターフェイス54を含む、コプロセッサ10内の接続を介してホストプロセッサ152に結合される。ホストプロセッサ152およびより特定のにはバスインターフェイスユニット40は、バックサイドL2インターフェイス52を介してCPUバス34におよびL2キャッシュ38に結合される。

#### 【0076】

図9の実施例は、エミュレーションコプロセッサ150とホストプロセッサ152との間のキャッシュ資源およびMMU資源の共有を見込む。言換えれば、エミュレーションコプロセッサ150は、この実施例ではキャッシュおよびMMU回路構成を排除するであろう。代わりに、エミュレーションコプロセッサ150には、Iキャッシュ44、Dキャッシュ46、MMU42、および間接的にバスインターフェイスユニット40へのアクセスが与えられるであろう。有利には、エミュレーションコプロセッサ150を実現するために採用される回路構成の量は、実質的に低減可能である。

#### 【0077】

なお、エミュレーションコプロセッサ150は、図2の前の記載と同様に、さまざまな実施例での、データキャッシュ46または命令キャッシュ44のいずれかからの命令をフェッチするよう構成可能である。さらになお、エミュレーションコプロセッサ150は、命令をフェッチするために命令キャッシュを含んでもよく、データキャッシュ46から命令キャッシュミスをフェッチしてもよい。

**【0078】**

プロセッサ10内にコマンドインターフェイス54を設ける代わりとして、FIFOを主メモリ14内に維持して、ホストプロセッサ152とエミュレーションコプロセッサ150との間でコマンドメッセージをやり取りしてもよい。なお、図9の実施例では、プロセッサ10は、とりわけ、単一の半導体基板、マルチチップモジュール、またはスロット1、スロットAもしくはスロット200タイプパッケージ内の2つ以上の半導体として実現されてもよい。

**【0079】**

図10を次に参照すると、コンピュータシステム5の第2の実施例のブロック図が示される。図3～図6は一般的に、この実施例にも適用可能である。他の実施例が可能であり企図される。図10の実施例では、ホストプロセッサ152およびエミュレーションコプロセッサ150は各々、バスブリッジ12に直接結合される。独立のCPUバス接続34および34aの代替として、ホストプロセッサ152およびエミュレーションコプロセッサ150は、共通のCPUバス34を共有してもよい。さらに、ホストプロセッサ152は、L2キャッシュ38に結合され、エミュレーションコプロセッサ150は同様に、L2キャッシュ38aに結合される。

**【0080】**

図10の実施例では、ホストプロセッサ152およびエミュレーションコプロセッサ150は各々、内部キャッシュおよびメモリ管理ファシリティを含んでもよい。たとえば、ホストプロセッサ152は、エミュレーションコプロセッサなしのコンピュータシステム内に含まれるよう設計されるプロセッサであってもよく、同様に、エミュレーションコプロセッサ150は、（たとえば、コンピュータシステムの中央処理ユニットとして）ホストコンピュータなしのコンピュータシステム内に含まれるよう設計されたプロセッサであってもよい。言換えれば、ホストプロセッサ152およびエミュレーションコプロセッサ150は、「既製（“off-the-shelf”）」の部品であってもよい。コマンドインターフェイス54が、FIFOを用いてプロセッサ間でコマンドメッセージをやり取りするなど、主メモリ14を介して設けられてもよい。代替的に、コマンドインターフェイ

ス54は、バスブリッジ12内に設けられてもよい。さらに別の代替として、CPUバス34および34aとは別個の専用インターフェイスを用いてコマンドインターフェイス54を設けてもよい。

#### 【0081】

図10の実施例では、ホストプロセッサ152およびエミュレーションコプロセッサ150は、論理的および電氣的に等価のバスインターフェイス（すなわちCPUバス34）を含む。図11は、エミュレーションコプロセッサ150がCPUバス34とは異なったバスインターフェイスを含むようなコンピュータシステム5の別の実施例である。したがって、図11に示すコンピュータシステム5は、エミュレーションコプロセッサ150によって生成されたトランザクションを、エミュレーションコプロセッサ150のバスインターフェイスのprotocolsおよび電氣的信号法特性から、CPUバス34aのそれに変換するためのバスブリッジ180を含む。したがって、図11の実施例は、たとえ異なったバスインターフェイスがホストプロセッサおよびエミュレーションコプロセッサによって用いられたとしても、既製のホストプロセッサ152および既製のエミュレーションコプロセッサ150をサポートする。

#### 【0082】

図10の実施例でのように、コマンドインターフェイス54は、図11の実施例において主メモリ内で実現されてもよい。代替的に、コマンドインターフェイス54は、バスブリッジ12内に設けられてもよい。さらに別の代替として、CPUバス34および34aとは別個の専用インターフェイスを用いてコマンドインターフェイス54を設けてもよい。

#### 【0083】

次に図12を参照すると、コンピュータシステム5の第4の実施例が示される。他の実施例が可能であり企図される。図12の実施例では、エミュレーションコプロセッサは、エミュレーションコプロセッサカード22C上に含まれる。エミュレーションコプロセッサカード22Cは、図12に示すとおりPCIバス24に結合される。エミュレーションコプロセッサカード22Cのためのハードウェアは、たとえば、カリフォルニア州のサニーベイルのリブライ社（Replay Cor

poration) によって製造されるラディウス・デタント (Radius Detente) AX またはMXカードであってもよい。

#### 【0084】

図3～図6に関して上述した操作に加えて、図12の実施例は、いくつかの他の操作をも含む。コマンドインターフェイスは、エミュレーションコプロセッサカード22C上のメモリ内に維持されてもよい。加えて、エミュレーションコプロセッサカード22CはI/Oデバイスであるため、エミュレーションコプロセッサカード22Cにインターフェイスするために、ドライバがオペレーティングシステム内に設けられる。さらになお、エグゼクティブソフトウェアがエミュレーションコプロセッサカード22Cについて設けられ、複数のアプリケーションプログラムが同時に実行中であることを可能にする。したがって、プロセスおよびプロセス内のスレッドを生成しかつ破壊するためのコマンドが、図6に例示されるようなエミュレーションコプロセッサとホストプロセッサとの間で交信され得るコマンドの組に加えられ得る。加えて、コマンドがページを割当てるために与えられ、エミュレーションコプロセッサカード22C上で実行する外部アプリケーションプログラムによって使用される。エミュレーションコプロセッサカード22Cは、コンピュータシステム5のオペレーティングシステムにとってはI/Oデバイスであるように見えるので、ページがエミュレーションコプロセッサに割当てられると、ページは主メモリ14にロックされる（すなわち、オペレーティングシステムによるページ割当て要求の受信の際にディスクドライブへのページアウトのためにページが選択されない）。エミュレーションコプロセッサカード上で実行するエグゼクティブは、いつページがエミュレーションコプロセッサ上で実行するアプリケーションプログラムによって使用されなくなるかを決定し、もはや使用されていないことを決定するとメッセージを与えてページをロック解除する。

#### 【0085】

さらに、エミュレーションコプロセッサカード22C内のエミュレーションコプロセッサは、1つ以上のキャッシュを含み、エミュレーションコプロセッサカード22C上で実行するエグゼクティブは、エミュレーションコプロセッサキャッ



シュと、ホストプロセッサ150およびL2キャッシュ38内（および、もし含まれていれば、ホストプロセッサ150aおよびL2キャッシュ38a内）のキャッシュとのキャッシュコヒーレンスを維持する。代替的に、エミュレーションコプロセッサ内のキャッシュは、キャッシュコヒーレンスが問題とならないように不能化されてもよい。

#### 【0086】

ある特定の実施例では、コンピュータシステム5は、アルファ（Alpha）命令セットアーキテクチャのためにウィンドウズNTオペレーティングシステムを採用し、ホストプロセッサ150は、アルファ命令セットアーキテクチャを採用する。さらに、コンピュータシステム5によって採用されるウィンドウズNTオペレーティングシステムは、Wx86エミュレーション拡張命令を含む。しかしながら、x86プロセッサをエミュレーションするためのコードは、上述したエミュレーションインターフェイスコードと置換えられる。エミュレーションコプロセッサカード22Cについてのドライバは、エグゼクティブからのロック要求およびロック解除要求に応答してページロックおよびロック解除機能を与える。より特定のには、エグゼクティブは、コードまたはデータのいずれかについてロックされたページを要求する。ドライバは、要求に応答して、ウィンドウズNTメモリマネージャ・アプリケーションプログラミングインターフェイス（API）コールを用いてページをロックする（すなわち、ページがディスクにスワップされることを防いで、異なった仮想ページがその物理的ページに割当てられることを可能にする）。その後、エグゼクティブは、ページがもはやアプリケーションプログラム実行のために必要とされていないこと決定し得り、ロック解除メッセージを送信し得る。これに応答して、ドライバは、ウィンドウズNTメモリマネージャAPIを用いてページをロック解除する。加えて、ドライバは、オペレーティングシステム内のカードを初期化しかつカード上にメモリをマッピングする責任がある。

#### 【0087】

図12の実施例についてのドライバおよびエグゼクティブは、ウィンドウズNTオペレーティングシステムによって定義される（より特定のには、ウィンドウ

ズNTDDKにドキュメントされるような) パケットベースのDMAバスマスタモデルを基礎とする。I o 割当てアダプタチャネル (Io Allocate Adapter Channel) を用いてアダプタオブジェクトが生成される。MDL (メモリ記述子リスト) が生成され、プロセスによって用いられるページの仮想から物理的へのマッピングを記述する。論理的アドレスは、I o マップ転送 (Io Map Transfer) で生成され、エミュレーションコプロセッサカード22C上のエミュレーションコプロセッサに与えられる。したがって論理的アドレスの、メモリ14 (すなわちホストシステムメモリ) 内の物理的アドレスへの変換を与えるマッピングレジスタが生成される。エミュレーションコプロセッサはこれによって、主メモリ14にアクセスして、命令および読出/書込データを直接フェッチすることが可能である。言換えれば、コードが実行され、データが主メモリ14内の適所にアクセスされる。これらのアクセスは、ホストシステムにとってはDMAのように見えるかもしれない。このようにして、命令およびデータは、エミュレーションコプロセッサカード22Cに与えられる。

#### 【0088】

なお、エグゼクティブソフトウェアが、エミュレーションコプロセッサカード22Cを制御するために記載されたが、制御の一部をハードウェアに設けるような他の実施例が可能である。そのような実施例が企図される。

#### 【0089】

次に図13を参照すると、エミュレーションコプロセッサカード22Cのある実施例のブロック図が示される。他の実施例が可能であり企図される。図13に示すとおり、エミュレーションコプロセッサカード22Cは、PCIインターフェイス190と、エミュレーションコプロセッサ150と、メモリ194とを含む。PCIインターフェイス190は、PCIバス24、メモリ194およびエミュレーションコプロセッサ150に結合される。エミュレーションコプロセッサ150はさらに、メモリ194に結合される。メモリ194は、エグゼクティブプログラム196と、エグゼクティブプログラム196とエミュレーションコプロセッサカード22cのためのドライバとの間でコマンドメッセージおよびエミュレーションインターフェイスコード92をやり取りするために用いられるコ

マンドキュー198とのための記憶装置を含む。言換えれば、コマンドキュー198は、コマンドインターフェイス54を含み得る。なお、命令およびデータは好ましくは、主メモリ14からエミュレーションコプロセッサ150によって直接にアクセスされるが、代替の実施例は、主メモリ14内のページから転送される命令およびデータをメモリ194内に同様に記憶してもよい。

#### 【0090】

上述したとおり、エミュレーションコプロセッサカード22Cは、リプライ社によって製造されるラディアス・デタントAXまたはMXカードであってもよい。これらの製品は、図13には示されないさらなるハードウェア特徴を含んでもよい。ハードウェア特徴は、カードがエミュレーションコプロセッサカード22Cとして用いられる際、所望のごとく使用されてもよいしまたは使用されなくてもよい。

#### 【0091】

次に図14を参照すると、エグゼクティブプログラム196のある実施例によって維持される制御構造を例示するブロック図が示される。他の実施例が可能であり企図される。図14の実施例では、制御構造は、エミュレーションコプロセッサカード22C内でアクティブであるプロセスのダブルでリンクされたリストであるプロセスリスト200を含む。たとえば、図14では、3つのプロセス202A、202Bおよび202Cがアクティブである。各プロセスは、1つ以上のスレッドを含み得る。たとえば、プロセス202Aは、スレッド204A、204Bおよび204Cを含む。同様に、プロセス202Bは、スレッド204D、204Eおよび204Fを含む。プロセス202Cは、スレッド204Gを含む。各プロセスにはさらに、プロセスに対応する命令およびデータが記憶されているメモリの1つ以上のページが割当てられてもよい。たとえば、プロセス202Aには、ページ206A、206Bおよび206Cが割当てられる。同様に、プロセス202Bには、ページ206Dおよび206Eが割当てられる。プロセス202Cには、ページ206Fおよび206Gが割当てられる。

#### 【0092】

図14に例示されるとおり、各プロセス202A～202Cには、他のプロセ

ス202A～202Cとは異なった数のページ206および異なった数のスレッド204が割当てられてもよい。プロセス202A～202Cがページフォルトを経験すると、新しいページが、コマンドキュー198を介してページを要求するエグゼクティブプログラム196を介してそのプロセスに割当てられ得る。ページ要求は、ページがロックされるべきであるという表示を含む。そのページ内の処理が完了するとプロセス202A～202Cはページを明示開放することが可能となり（たとえば動的に割当てられるメモリ）、ここでエグゼクティブプログラム196はページロック解除メッセージを伝送可能となる。さらに、ページは、プロセス内の特定のスレッドと関連付けられてもよい。そのようなページは、ページが関連付けられるスレッドの脱出の際に、開放され得る。加えて、プロセスが破壊されると、エグゼクティブプログラム196は、そのプロセスに割当てられる各ページについてページロック解除メッセージを伝送し得る。

#### 【0093】

エグゼクティブプログラム196は、プロセス生成および破壊コマンドメッセージのためにコマンドキュー198内にグローバルメッセージキューを維持してもよく、各スレッドごとにコマンドキュー198内にそのスレッドのためのコマンドメッセージを含むメッセージキューを維持してもよい。このようにして、エグゼクティブプログラムは、複数プロセス、複数スレッドのアプリケーションを処理するよう構成可能である。スレッド専用コマンドメッセージは、各スレッドごとに、ページロックおよびロック解除メッセージならびに生成および破壊メッセージを含み、各スレッドごとにゴーおよびストップメッセージを含み得る。したがって、プロセススケジューリングは、コンピュータシステム5上で実行するオペレーティングシステムによって処理可能である。ゴーメッセージおよびストップメッセージは、スケジューリングを実行するために使用され得る。さらに、表140に示すレジスタ読出および書込コマンドは、1スレッドずつのベースで与えられてもよい。

#### 【0094】

図15を次に参照すると、エグゼクティブプログラム196のある実施例の操作を例示する1組のフローチャートが示される。他の実施例が可能であり企図さ

れる。図15の実施例では、第1のフローチャート220は、コプロセッサカード22Cのリセットを例示し、第2のフローチャート222は、休止プロセスを例示し、第3のフローチャート224は、エグゼクティブプログラム196の他の局面を例示する。

#### 【0095】

コプロセッサカード22Cがリセットされると（たとえば、コンピュータシステム5がブートされると）、フローチャート220が実行される。エグゼクティブプログラム196は、コプロセッサカード22C上の環境を初期化する（ステップ226）。たとえば、エグゼクティブプログラム196は、メモリ194を既知の状態にクリアし、エミュレーションコプロセッサ150によって使用されるページテーブルを生成し（および、エグゼクティブプログラム196自身によって使用されるエントリを初期化する）、コマンドキュー198を生成することが可能である。休止プロセスを初期化した後（ステップ228）、リセット処理が完了する。フローチャート222によって例示されるように、休止プロセスは、（たとえば、コマンドキュー198内のメッセージの受信によって）割込みされるまで何もしない（ステップ230）。

#### 【0096】

フローチャート224は、プロセスがコプロセッサカード22C内でアクティブである間のエグゼクティブプログラム196の動作を例示する。フローチャート224は、エグゼクティブプログラム196を呼出し得るさまざまな事象に依存して、いくつかのエントリポイント232、234および236を含む。

#### 【0097】

エントリポイント232は、メッセージがドライバによってコマンドキュー198に与えられると生じる。メッセージの受信によってエミュレーションコプロセッサ150の割込みが起き、この時点でエグゼクティブプログラム196が呼出される。割込みによって呼出されると、エグゼクティブプログラム196は、受取ったメッセージを処理する（ステップ238）。さまざまなメッセージが受取られる可能性がある。たとえば、プロセス生成またはスレッド生成メッセージが受取られるであろう。プロセス生成メッセージを受取ると、エグゼクティブプ

ログラム196は、プロセス202をプロセスリスト200に加える。同様に、スレッド生成メッセージを受取ると、エグゼクティブプログラム196は、スレッド204を、スレッド生成メッセージを受取ったプロセス202に加える。コンテキスト読出メッセージ（たとえばレジスタ読出コマンド）が、そのプロセス（および／またはスレッド）と関連付けられるコンテキストデータ構造からレジスタを読出し、要求された情報で応答メッセージを生成することによって、エグゼクティブプログラム196によって処理される。コンテキスト書込メッセージ（たとえばレジスタ書込コマンド）が、選択されたコンテキストデータ構造に値を書込むことによって、エグゼクティブプログラム196によって処理される。エグゼクティブプログラム196は、ゴーメッセージに応答してレディタスクのリストにスレッドを加え、ストップメッセージに応答してレディタスクのリストからスレッドを除去する。ページロック完了メッセージ（エグゼクティブプログラム196によって先に送られたページロックメッセージに応答して発行された）が、ロックされたページについての変換でページテーブルを更新し、かつページフォルトを経験したスレッドをレディタスクのリストに加えることによって、エグゼクティブプログラム196によってサービスされる。

#### 【0098】

メッセージを処理した後、エグゼクティブプログラム196は、レディタスクのリストからタスクを選択し、選択されたタスクに戻る（ステップ240）。

#### 【0099】

エントリポイント234は、ページフォルトが、エミュレーションコプロセッサ150によって実行されたタスク（たとえばプロセススレッド）によって経験されると生じる。ページフォルトに응答して、エグゼクティブプログラム196は、コマンドキュー198を介してページロックメッセージを送信する（ステップ242）。ページフォルトを経験したタスクは、ページロック完了メッセージがページについて受取られるまで、レディタスクのリストから除去される。上述したように、ページロック完了メッセージの受信によってタスクはレディタスクのリストに加えられる。その後、エグゼクティブプログラム196は、レディタスクのリストからタスクを選択し、選択されたタスクに戻る（ステップ240）。

）。

#### 【0100】

エントリポイント236は、違法操作コードトラップ例外がエミュレーションコプロセッサ150によって経験されると生じる。予め定義された違法操作コードを用いて、サンクが入力されたことを合図する（しばしは「BOP」と呼ばれる）。エグゼクティブプログラム196は、予め定義された違法操作コードが検出されたかどうかを決定する（決定ブロック244）。予め定義された違法操作コードが検出されなければ、例外メッセージがコマンドキュー198を介して送信され、違法操作コード例外を生成するタスクについて違法操作コード例外が送信されたことをオペレーティングシステムに通知する（ステップ246）。予め定義された違法操作コードが検出されなければ、ストップメッセージが送信され、ホストコードへの遷移のためにタスクが停止したことをオペレーティングシステムに通知する（ステップ248）。いずれの場合でも、例外を経験したタスクは、レディタスクのリストから除去され、レディタスクが、レディタスクのリストから選択される（ステップ240）。

#### 【0101】

##### 産業上の応用可能性

この発明は、コンピュータシステム内に応用可能である。上記の開示に従って、ある命令セットアーキテクチャを採用するエミュレーションコプロセッサを用いて、このために外部アプリケーションプログラムが設計されるが、第2の命令セットアーキテクチャに従ってコード化されているオペレーティングシステムを採用するコンピュータシステム内のその命令セットアーキテクチャ内にコード化される外部アプリケーションプログラムを実行するような、コンピュータシステムが示された。有利には、コンピュータシステムによって実行可能であるアプリケーションプログラムの数は増大する。加えて、アプリケーションプログラムの性能は、ソフトウェアエミュレーションおよび／またはバイナリ変換を用いて達成可能であるよりも実質的により優れているであろう。さらになお、エミュレーションコプロセッサはアーキテクチャを実現するので、アーキテクチャ的特異性をモデリングすることが排除される。結果として得られるコンピュータシステム

は、異種のマルチプロセッシングコンピュータシステムを形成する。

【0102】

以上の開示を完全に理解すると、多くの変形および変更が当業者には明らかとなるであろう。前掲の特許請求の範囲は、そのようなすべての変形および変更を含むものと解釈されるべきであることが意図される。

【図面の簡単な説明】

【図1】 コンピュータシステムのある実施例のブロック図である。

【図2】 ホストプロセッサコアおよびエミュレーションコプロセッサコアを含む図1に示すプロセッサのある実施例のブロック図である。

【図3】 外部アプリケーションをエミュレーションするプロセスのブロック図である。

【図4】 図1に示すコンピュータシステム内のアプリケーションプログラムの初期化のある実施例を例示するフローチャートである。

【図5】 図3に示すエミュレーションインターフェイスの呼出のある実施例を例示するフローチャートである。

【図6】 図1に示すプロセッサのある実施例に従う交信コマンドを例示する表である。

【図7】 図1に示すプロセッサの第2の実施例のブロック図である。

【図8】 図7に示すインターフェイス論理ブロックのある実施例の操作を例示するフローチャートである。

【図9】 図1に示すプロセッサの第3の実施例のブロック図である。

【図10】 コンピュータシステムの第2の実施例のブロック図である。

【図11】 コンピュータシステムの第3の実施例のブロック図である。

【図12】 コンピュータシステムの第4の実施例のブロック図である。

【図13】 図12に示すエミュレーションコプロセッサカードのある実施例のブロック図である。

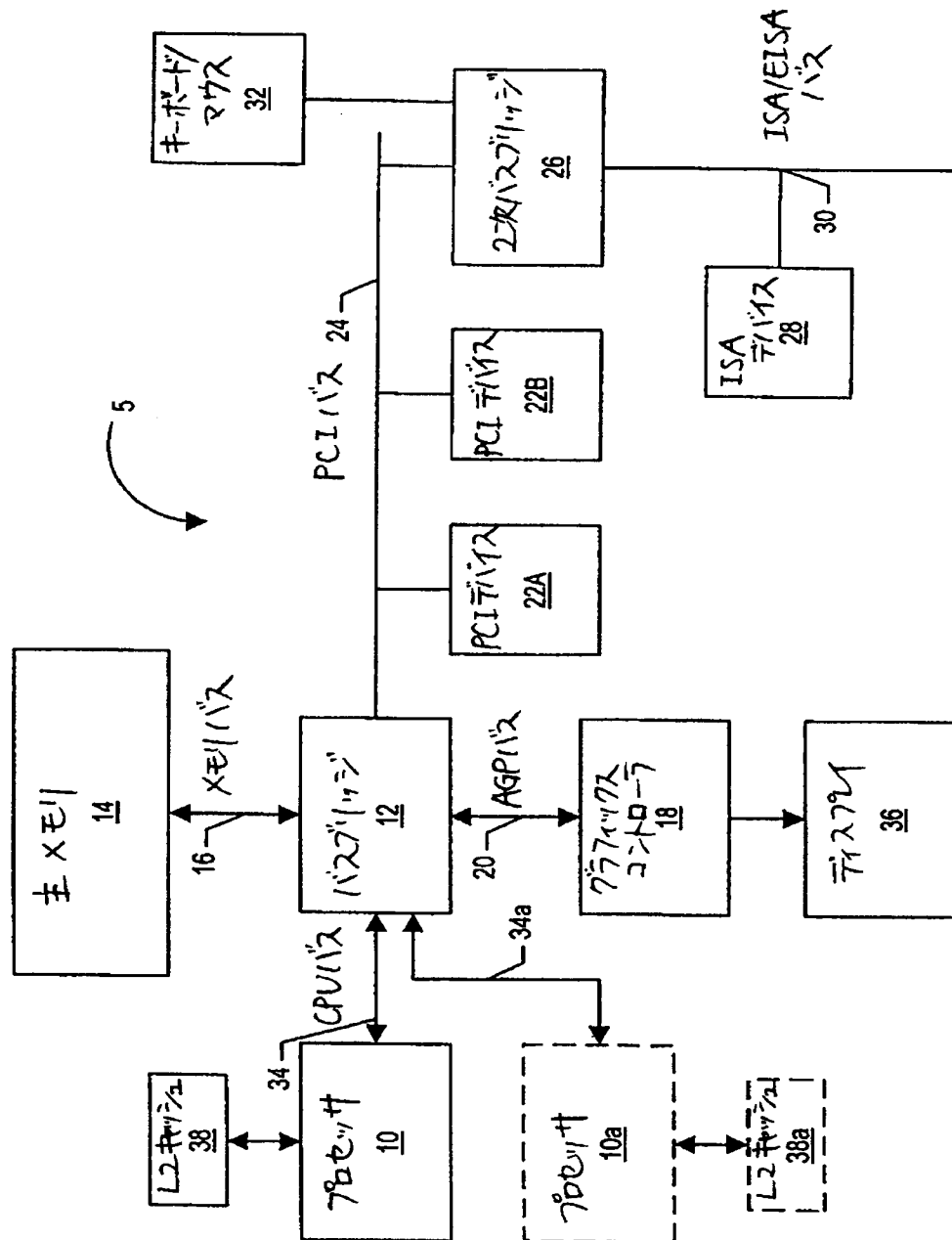
【図14】 図13に示すエグゼクティブプログラムのある実施例によって維持される制御構造を例示する図である。

【図15】 図13に示すエグゼクティブプログラムのある実施例を例示す

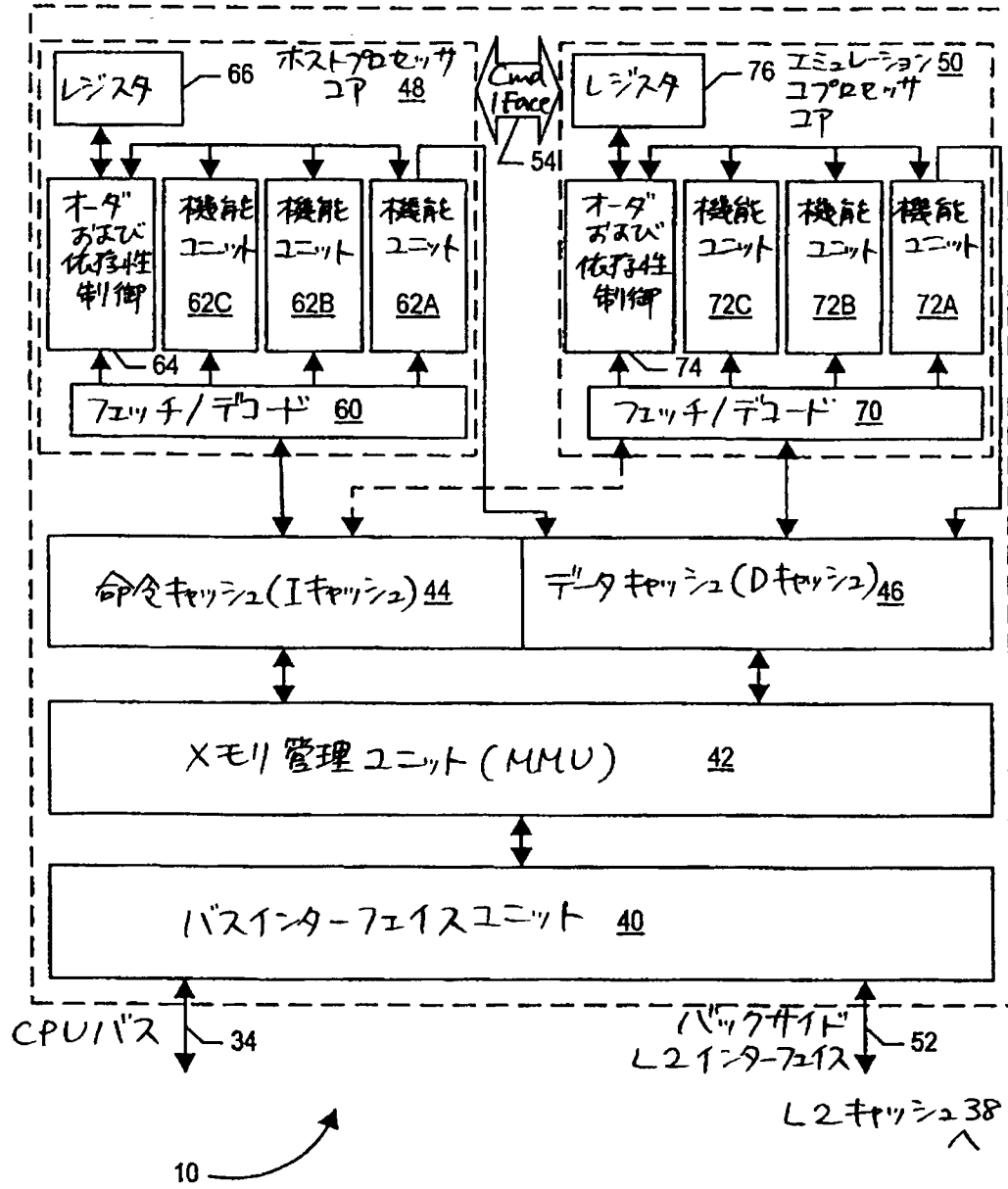


る1組のフローチャートである。

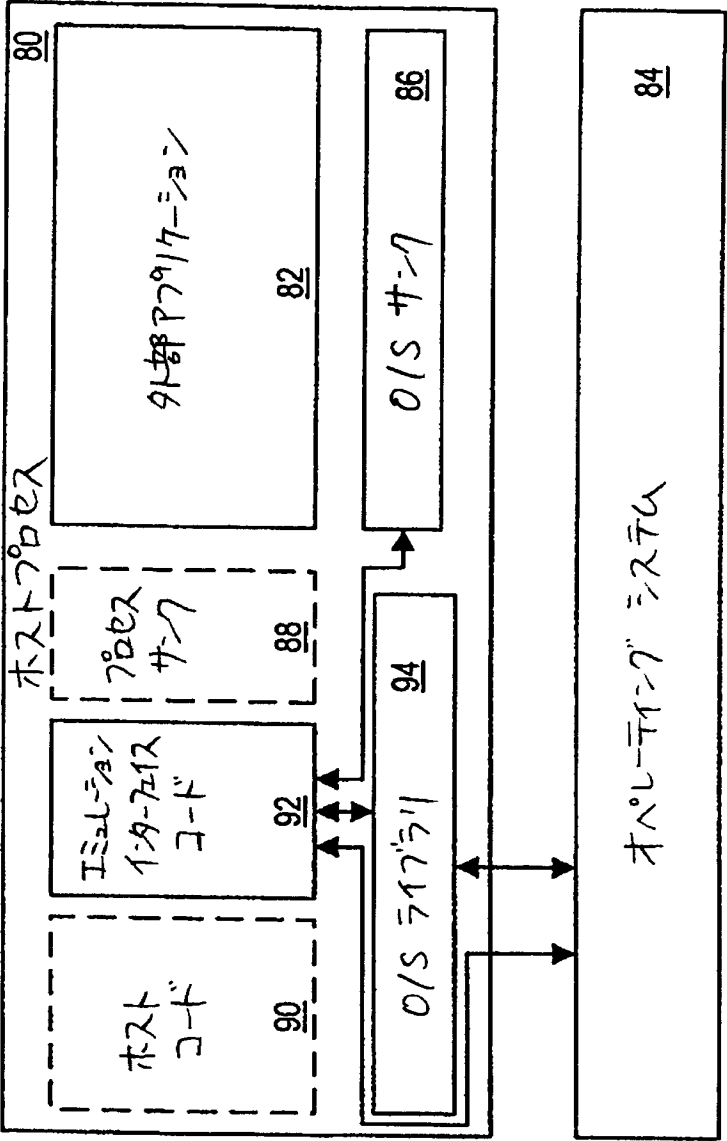
【図1】



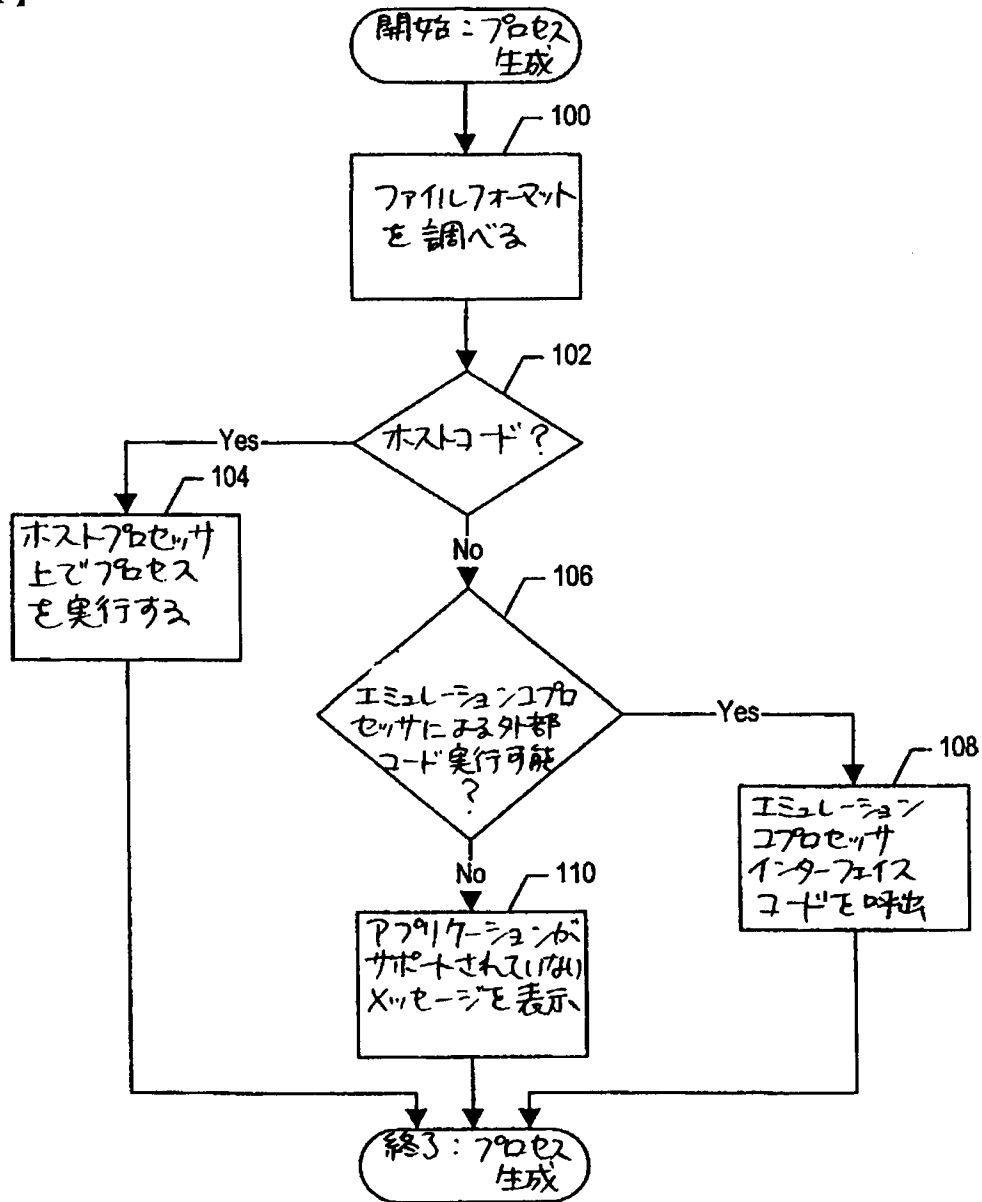
【図2】



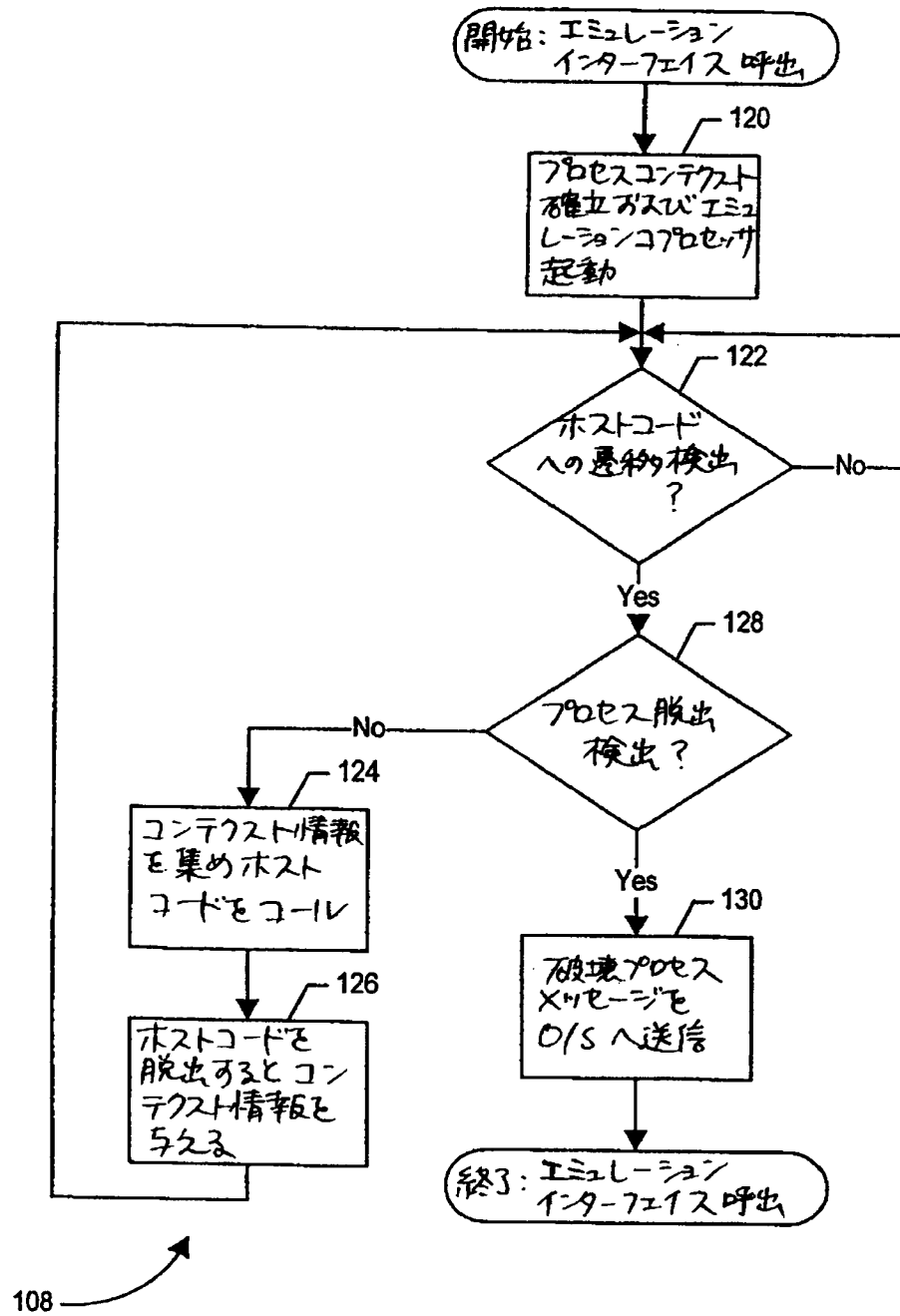
【図3】



【図4】



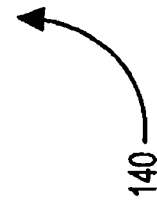
【図5】



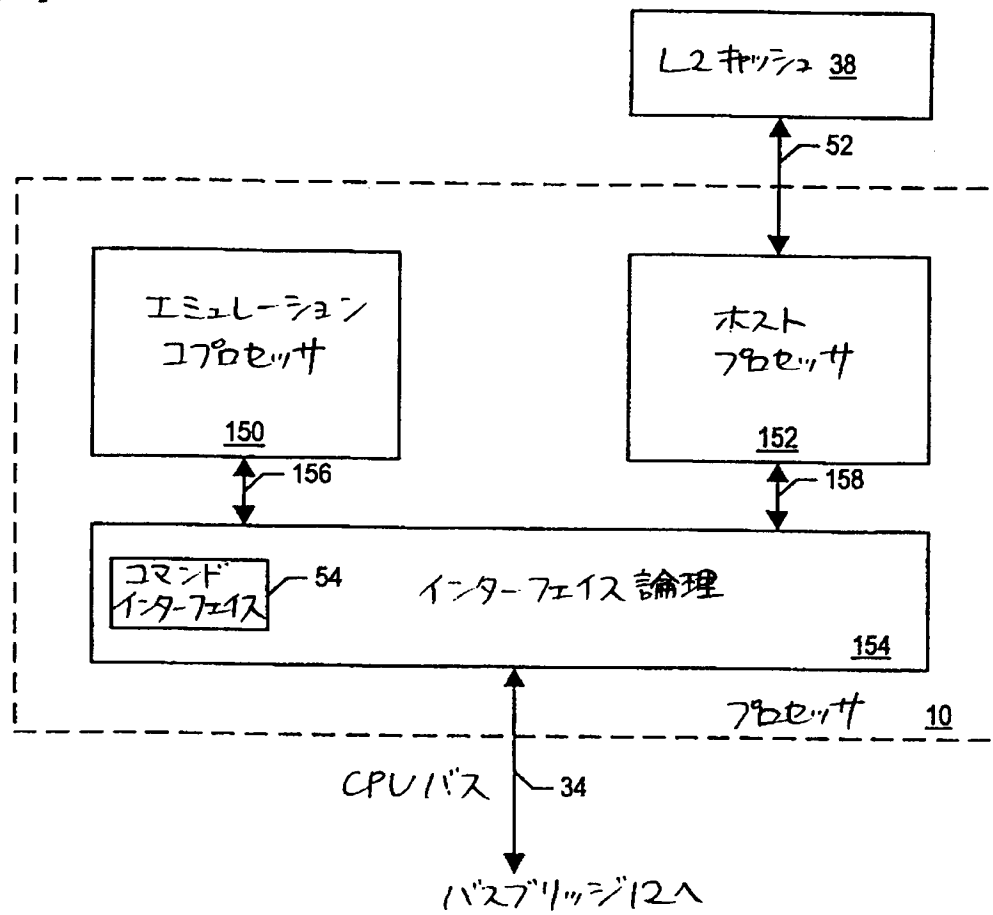
【図6】

コマンド	機能
レジスタ読出	1つ以上のエミュレーション/エプロセッサレジスタを 読出
レジスタ書込	1つ以上のエミュレーション/エプロセッサレジスタに 書込
ゴー	プログラムのカウンタ内に記録される実行ポインタ の実行を開始
ストップ	(停止の理由を含む)エミュレーション/エプロセッサが 停止したことをホストに通知

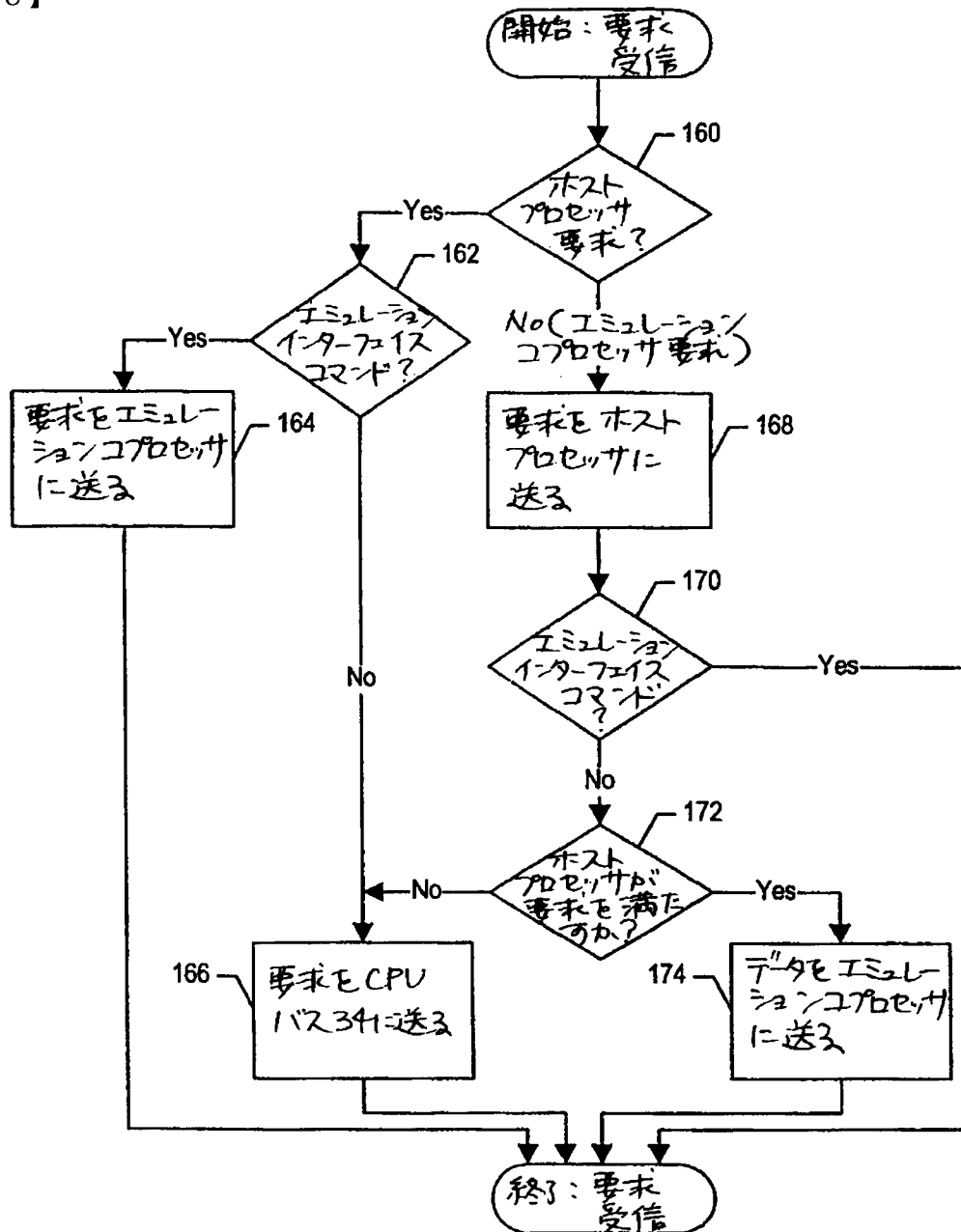
140



【図7】

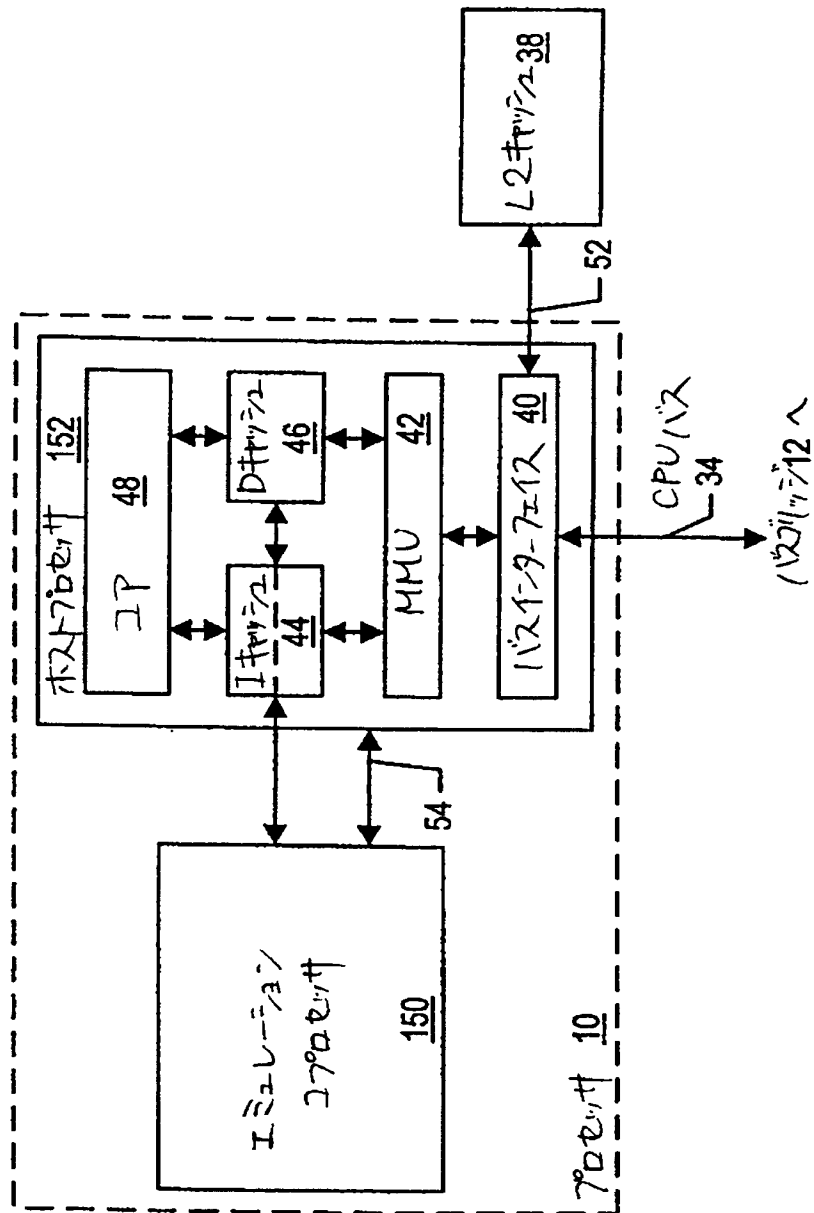


【図8】

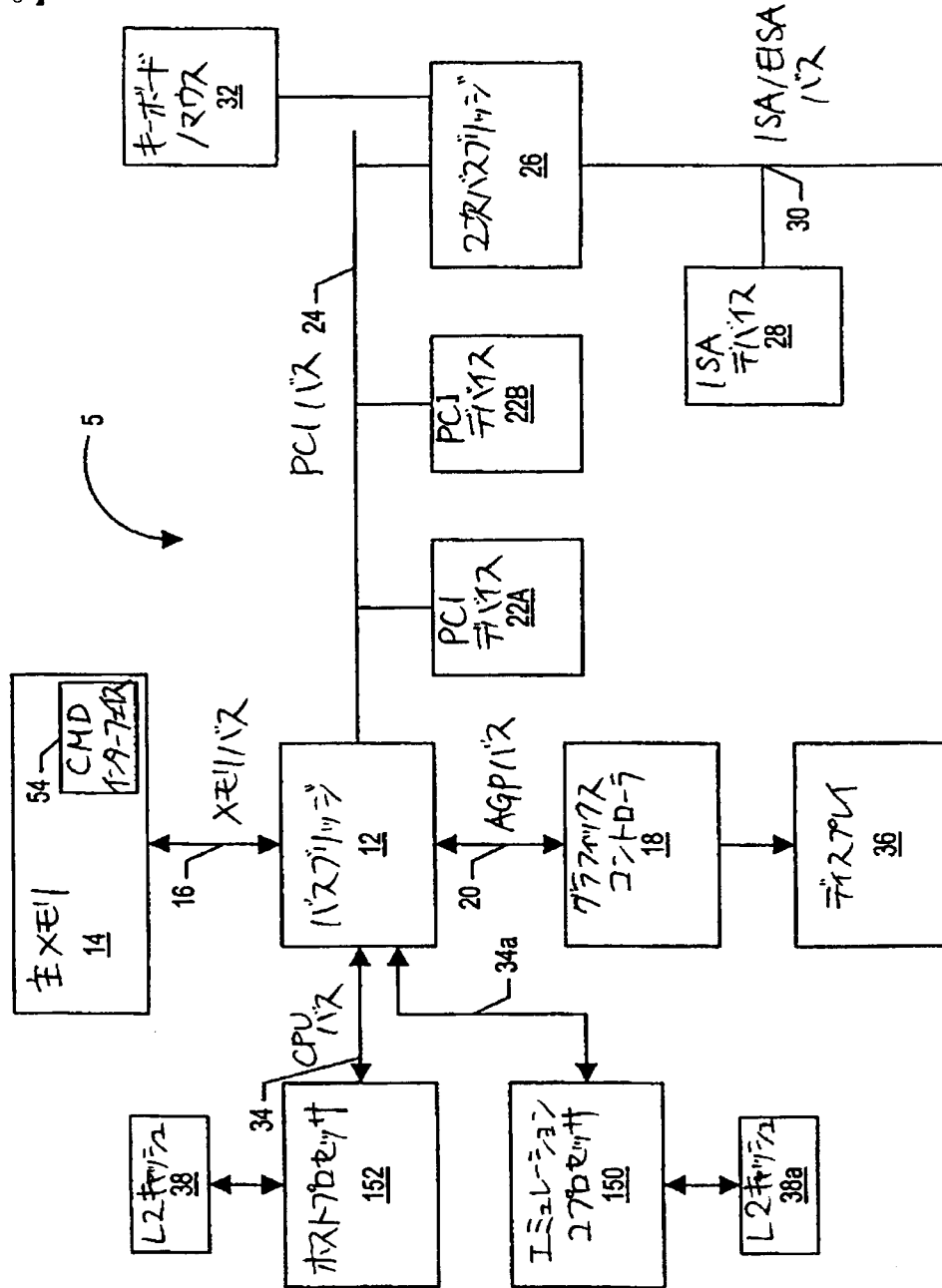




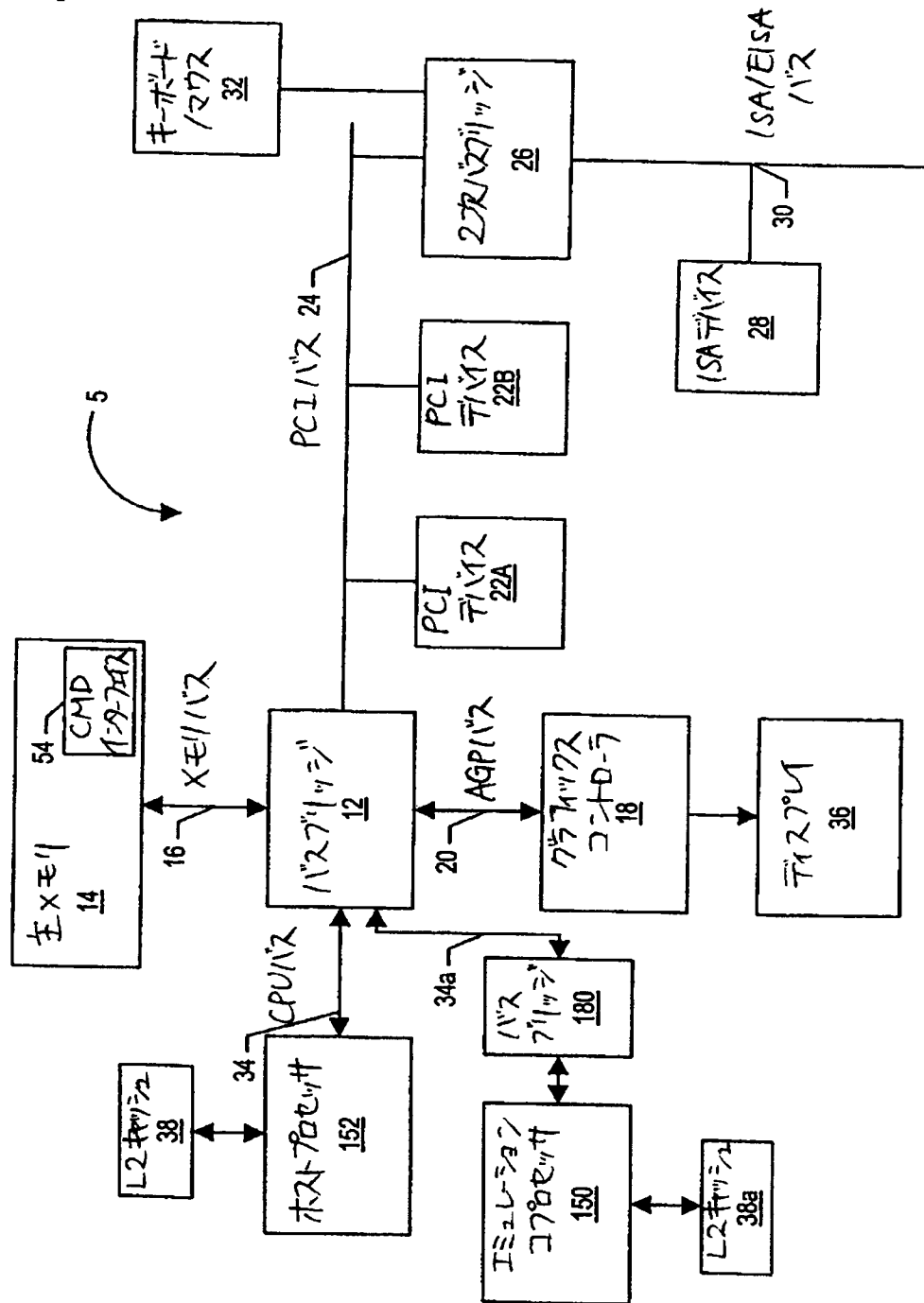
【図9】



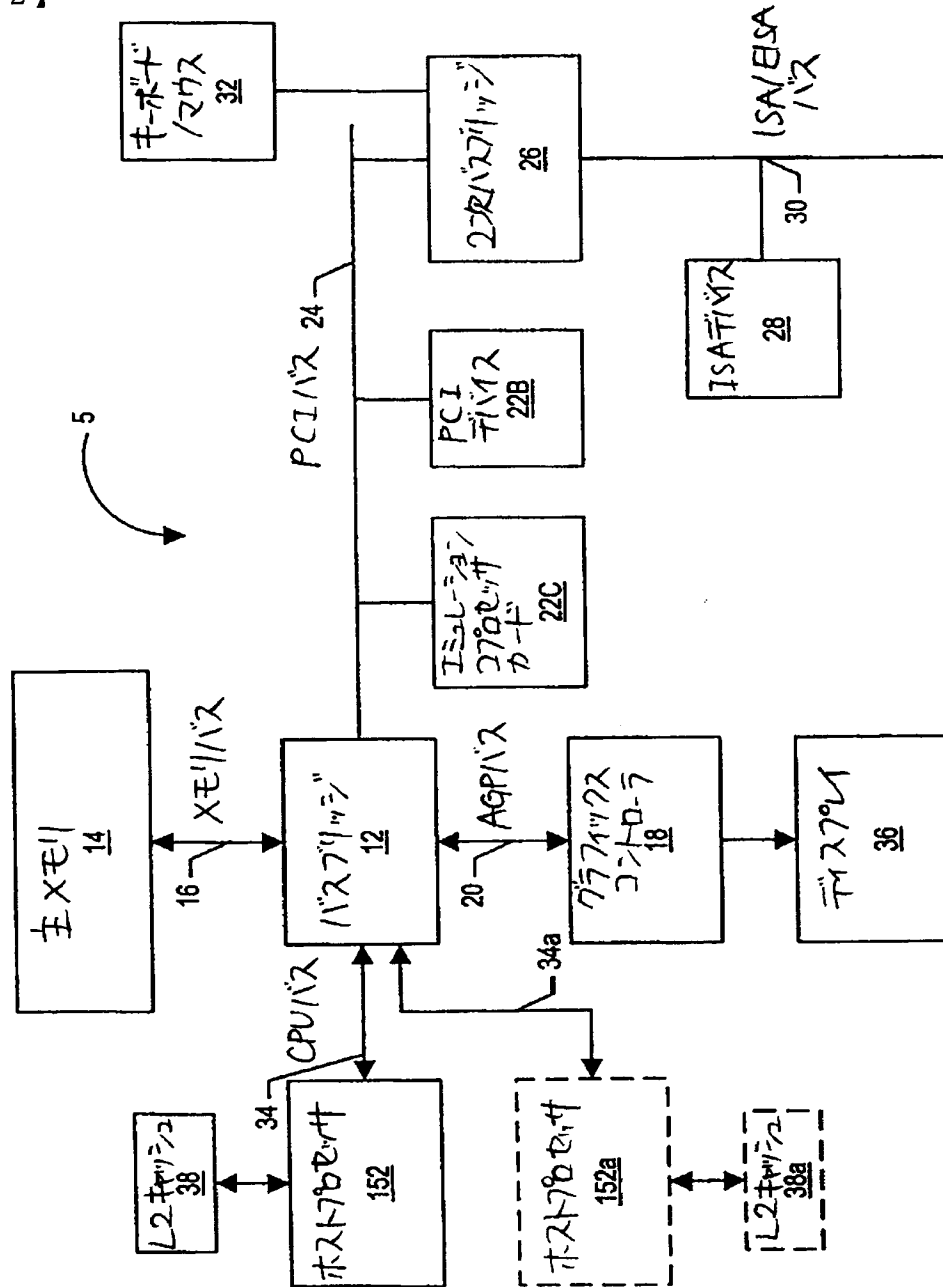
【図10】



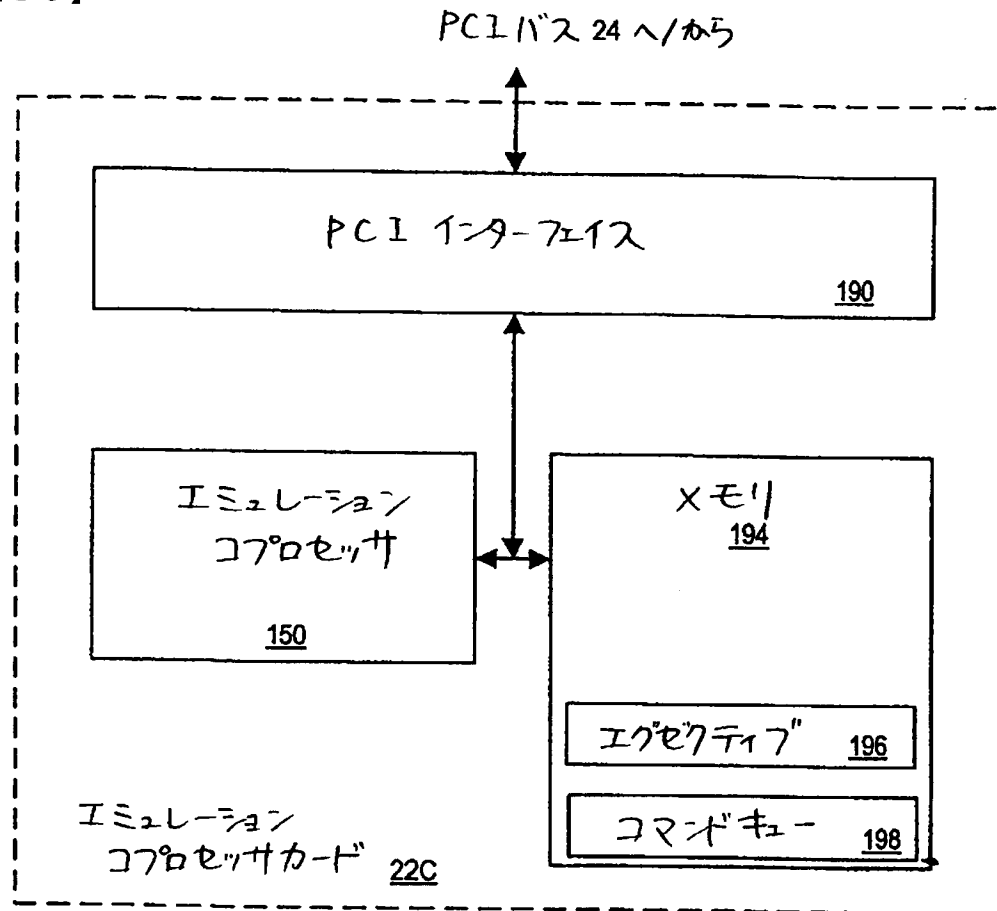
【図 1 1】



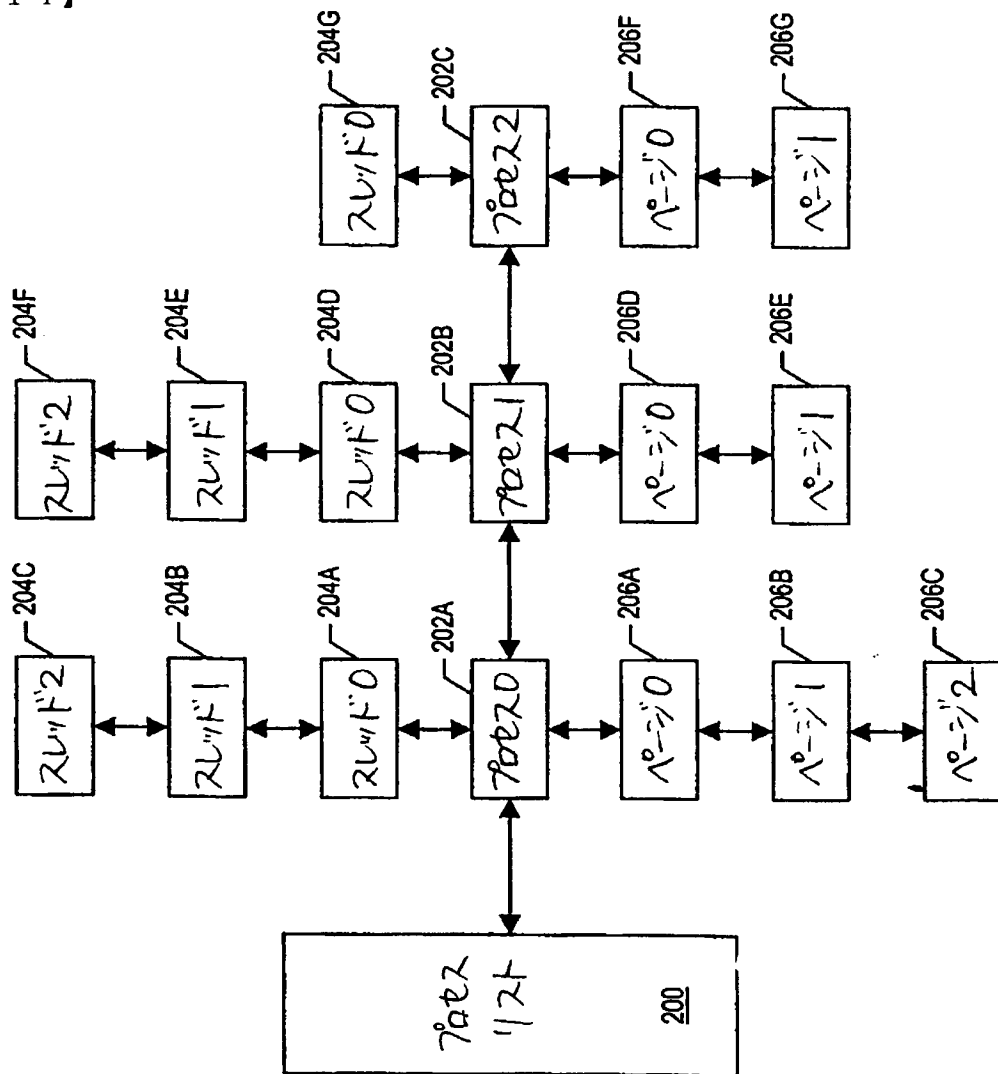
【図12】



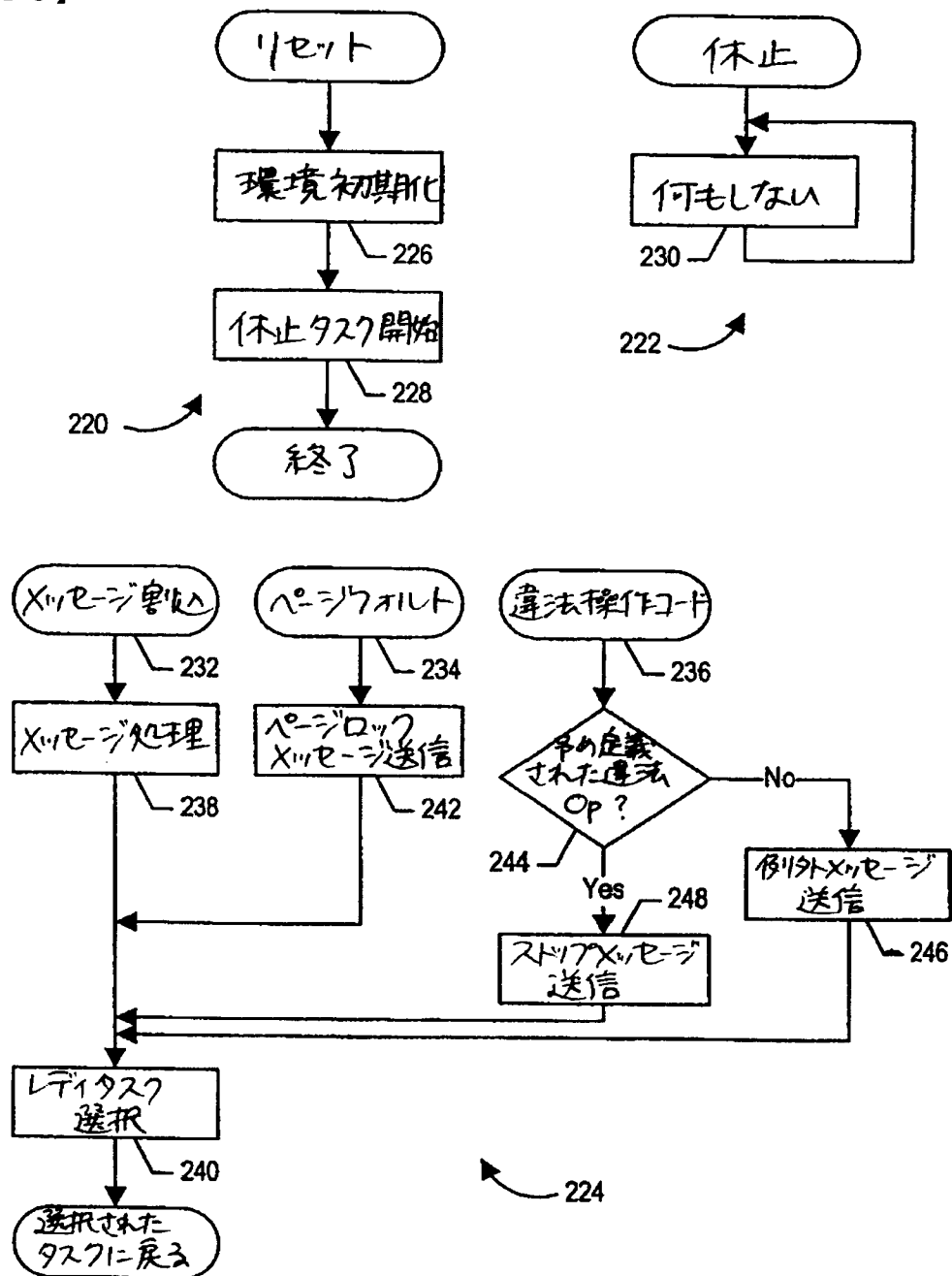
【図13】



【図14】



【図15】



【手続補正書】 特許協力条約第34条補正の翻訳文提出書

【提出日】 平成12年5月22日 (2000. 5. 22)

【手続補正1】

【補正対象書類名】 明細書

【補正対象項目名】 特許請求の範囲

【補正方法】 変更

【補正内容】

【特許請求の範囲】

【請求項1】 コンピュータシステムのための装置であって、

第1の命令セットアーキテクチャによって定義される第1の命令を実行するよう構成される第1のプロセッサ(48、152)を含み、前記コンピュータシステム(5)によって採用されるオペレーティングシステム(84)は、前記第1の命令を用いてコード化され、さらに、

前記第1のプロセッサ(48、152)に結合される第2のプロセッサ(50、150)を含み、前記第2のプロセッサ(50、150)は、前記第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャによって定義される第2の命令を実行するよう構成され、

前記第1のプロセッサ(48、152)は、前記オペレーティングシステム(84)を実行するよう構成され、

前記オペレーティングシステムは、前記オペレーティングシステム(84)内で実行するよう設計されるアプリケーションプログラム(82)を起動するためにユーザコマンドに応答して実行されるルーチンを含み、前記ルーチンは、前記アプリケーションプログラム(82)のファイルフォーマットを分析してアプリケーションプログラム(82)がどの命令セットアーキテクチャにコード化されるかを決定し、前記アプリケーションプログラム(82)が前記第2の命令セットアーキテクチャにコード化される場合、前記第2のプロセッサ(50、150)は、前記アプリケーションプログラム(82)を実行するよう構成され、前記第2のプロセッサ(50、150)は、前記アプリケーションプログラム(82)のためのオペレーティングシステムルーチン(94)の使用を検出すると前記



第1のプロセッサ(48、152)と通信するよう構成されることを特徴とする、コンピュータシステムのための装置。

【請求項2】 前記第1のプロセッサ(48)および前記第2のプロセッサ(50)は、1つ以上のキャッシュ(44、46)に結合され、前記第1のプロセッサ(48)および前記第2のプロセッサ(50)は、前記1つ以上のキャッシュ(44、46)を共有するよう構成される、請求項1に記載の装置。

【請求項3】 前記第1のプロセッサ(48)および前記第2のプロセッサ(50)は、1つ以上のメモリ管理ユニット(42)に結合され、前記第1のプロセッサ(48)および前記第2のプロセッサ(50)は、前記メモリ管理ユニット(42)を共有するよう構成される、請求項2に記載の装置。

【請求項4】 前記第1のプロセッサ(48、152)および前記第2のプロセッサ(50、150)は、予め定められた制御プロトコルを介して通信するよう構成され、前記制御プロトコルは、前記第1のプロセッサ(48、152)と前記第2のプロセッサ(50、150)との間でやり取りされるメッセージを含み、前記メッセージは、前記コンピュータシステム(5)内のメモリ(14)を介してやり取りされる、請求項1、2または3に記載の装置。

【請求項5】 前記第2のプロセッサ(50、150)は、特定の違法操作コードを実行することによって前記オペレーティングシステムルーチンの前記使用を検出するよう構成される、請求項1、2、3または4に記載の装置。

【請求項6】 前記第1のプロセッサ(48、152)は、前記第2のプロセッサ(50、150)からコンテキスト情報を要求し、前記オペレーティングシステムルーチン(94)を実行し、前記第2のプロセッサ(50、150)との通信によって前記第2のプロセッサ(50、150)に前記アプリケーションプログラム(82)の制御を戻すよう構成され、前記コンテキスト情報は、前記第2のプロセッサ(50、150)内のレジスタからの少なくとも1つのレジスタ値を含む、請求項5に記載の装置。

【請求項7】 前記アプリケーションプログラムが前記第1の命令セットアーキテクチャにコード化された場合、前記第1のプロセッサ(48、152)は、前記アプリケーションプログラムを実行するよう構成される、請求項1～6の

いずれかに記載の装置。

【請求項8】 第1の命令セットアーキテクチャからの命令を用いてコード化され、前記第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャからの命令を用いてコード化されるオペレーティングシステム（84）内で実行するよう設計されるアプリケーションプログラム（82）を実行するための方法であって、

前記アプリケーションプログラム（82）が起動されていることを検出するステップを含み、前記検出は、前記第2の命令セットアーキテクチャからの命令を実行するよう構成される第1のプロセッサ（48、152）上で実行する前記オペレーティングシステム（84）によって行なわれ、さらに、

前記第1の命令セットアーキテクチャからの命令を実行するよう構成される第2のプロセッサ（50、150）内の前記アプリケーションプログラム（82）のためのコンテキストを確立するステップと、

前記第2のプロセッサ（50、150）上で前記アプリケーションプログラム（82）を実行するステップとを含み、

前記検出は、前記アプリケーションプログラムのファイルフォーマットを分析して前記アプリケーションプログラム（82）がどの命令セットアーキテクチャにコード化されるかを決定するステップを含み、前記確立は、前記アプリケーションプログラムが前記第1の命令セットアーキテクチャにコード化されることを検出することに応答することを特徴とする、方法。

【請求項9】 前記オペレーティングシステム（84）内のオペレーティングシステムルーチン（94）への前記アプリケーションプログラム（82）内の遷移を検出するステップと、

前記第1のプロセッサ（48、150）上で前記オペレーティングシステムルーチン（94）を実行するステップと、

前記オペレーティングシステムルーチン（94）を前記実行した後に前記第2のプロセッサ（50、150）上で実行する前記アプリケーションプログラム（82）に戻るステップとをさらに含む、請求項8に記載の方法。

【請求項10】 前記アプリケーションプログラムが前記第2の命令セット

アーキテクチャにコード化されることを検出することに応答して前記第1のプロセッサ(48、152)上で前記アプリケーションプログラム(82)を実行するステップをさらに含む、請求項8または請求項9に記載の方法。

【請求項11】 異種のマルチプロセッシングシステムであって、

第1の命令セットアーキテクチャによって定義される第1の命令を実行するよう構成される第1のプロセッサ(48、152)と、

前記第1のプロセッサ(48、152)に結合される第2のプロセッサ(50、150)とを含み、前記第2のプロセッサ(50、150)は、前記第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャによって定義される第2の命令を実行するよう構成され、さらに、

前記第1の命令を用いてコード化されるオペレーティングシステム(84)と、

前記第2の命令セットを用いてコード化されかつ前記オペレーティングシステム(84)内で実行するよう設計されるアプリケーションプログラム(82)とを含み、

前記第2のプロセッサ(50、150)は、前記アプリケーションプログラム(82)を実行するよう構成され、前記第1のプロセッサ(48、152)は、前記アプリケーションプログラム(82)と関連しないプロセス(80)を同時に実行するよう構成される、異種のマルチプロセッシングシステム。

【請求項12】 前記第2のプロセッサ(50、150)は、実行中に前記アプリケーションプログラム(82)によって前記オペレーティングシステム(84)内のオペレーティングシステムルーチン(94)の使用を検出するよう構成される、請求項11に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項13】 前記第2のプロセッサ(50、150)は、特定の違法操作コードを実行することによって前記使用を検出するよう構成される、請求項12に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項14】 前記第2のプロセッサ(50、150)は、前記使用を検出すると前記第1のプロセッサ(48、152)と交信するよう構成される、請

求項12に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項15】 前記第1のプロセッサ(48、152)は、前記第2のプロセッサ(50、150)からコンテキスト情報を要求し、前記オペレーティングシステムルーチン(94)を実行し、前記第2のプロセッサ(50、150)との通信によって前記アプリケーションプログラム(82)の制御を前記第2のプロセッサ(50、150)に戻すよう構成される、請求項14に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項16】 前記アプリケーションプログラム(82)と関連しない前記プロセス(80)は、第2のアプリケーションプログラムを含む、請求項11に記載の異種のマルチプロセッシングコンピュータシステム。

【請求項17】 第1の命令セットアーキテクチャからの命令を用いてコード化され、前記第1の命令セットアーキテクチャと異なる第2の命令セットアーキテクチャからの命令を用いてコード化されるオペレーティングシステム(84)内で実行するよう設計されるアプリケーションプログラム(82)を実行するための方法であって、

前記アプリケーションプログラム(82)が起動されていることを検出するステップを含み、前記検出は、前記第2の命令セットアーキテクチャからの命令を実行するよう構成される第1のプロセッサ(48、152)上で実行する前記オペレーティングシステム(84)によって行なわれ、さらに、

前記第1の命令セットアーキテクチャからの命令を実行するよう構成される第2のプロセッサ(50、150)内の前記アプリケーションプログラム(82)のためのコンテキストを確立するステップと、

前記第2のプロセッサ(50、150)上で前記アプリケーションプログラム(82)を実行するステップとを含む、方法。

【請求項18】 前記オペレーティングシステム(84)内のオペレーティングシステムルーチン(94)への前記アプリケーションプログラム(82)内の遷移を検出するステップをさらに含む、請求項17に記載の方法。

【請求項19】 前記第1のプロセッサ(48、152)上で前記オペレーティングシステムルーチン(94)を実行するステップをさらに含む、請求項1

8に記載の方法。

【請求項20】 前記オペレーティングシステムルーチン（94）を前記実行した後に前記第2のプロセッサ（50、150）上で実行する前記アプリケーションプログラム（82）に戻るステップをさらに含む、請求項19に記載の方法。

## 【国際調査報告】

## INTERNATIONAL SEARCH REPORT

International Application No.  
PCT/US 99/01456

A. CLASSIFICATION OF SUBJECT MATTER  
IPC 6 G06F9/38

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)  
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 455 345 A (MATSUSHITA ELECTRIC WORKS LTD) 6 November 1991	11, 17
Y	see page 3, lines 9-51; page 4, lines 30-33; page 5, lines 25-58; page 7, line 13 - page 8, line 10; page 9, lines 12-14; page 10, line 16 - page 11, line 41	1, 5-8, 10, 12, 14, 18-20
Y	US 4 799 150 A (BUI LE) 17 January 1989  see column 1, line 39 - column 3, line 27; column 3, lines 48-50	1, 5-8, 10, 12, 14, 18-20
X	EP 0 817 096 A (TEXAS INSTRUMENTS INC) 7 January 1998	11, 16
A	see the whole document	1-3, 5, 12, 17
-/-		

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

## \* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"Z" document member of the same patent family

Date of the actual completion of the international search

3 June 1999

Date of mailing of the international search report

09/06/1999

Name and mailing address of the ISA

European Patent Office, P.O. 5818 Patentkanal 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax (+31-70) 340-3016

Authorized officer

Klocke, L

## INTERNATIONAL SEARCH REPORT

Inter-  
national Application No.  
PCT/US 99/01456

C.(Continued) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 230 353 A (IBM) 29 July 1987 see column 2, line 46 - column 3, line 6 ----	1,11,17
A	WO 93 16437 A (APPLE COMPUTER) 19 August 1993  see the whole document -----	1,2,7,9, 11,12, 17-20
A	US 5 077 657 A (COOPER THAYNE C ET AL) 31 December 1991 see the whole document -----	1,11,17

## INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter national Application No

PCT/US 99/01456

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0455345 A	06-11-1991	JP 2834837 B	14-12-1998
		JP 3282904 A	13-12-1991
		DE 69126166 D	26-06-1997
		DE 69126166 T	04-12-1997
		KR 9503552 B	14-04-1995
		US 5371860 A	06-12-1994
US 4799150 A	17-01-1989	NONE	
EP 0817096 A	07-01-1998	JP 10083304 A	31-03-1998
EP 0230353 A	29-07-1987	US 4787026 A	22-11-1988
		BR 8700173 A	01-12-1987
		JP 62171066 A	28-07-1987
WO 9316437 A	19-08-1993	AU 3616793 A	03-09-1993
		DE 4390577 T	23-02-1995
		JP 7504054 T	27-04-1995
		US 5577250 A	19-11-1996
US 5077657 A	31-12-1991	JP 3087930 A	12-04-1991



---

フロントページの続き

(72)発明者   ゴリシエク・ザ・フォース, フランク・ジ  
              エイ

              アメリカ合衆国、78749   テキサス州、オ  
              ースティン、オアシス・ドライブ、6611

(72)発明者   ボスウェル, チャールズ・アール

              アメリカ合衆国、78759   テキサス州、オ  
              ースティン、モラド・サークル、10610

Fターム(参考) 5B013 DD03 DD05

              5B045 DD01 DD12 GG06 GG09

【要約の続き】

リケーションプログラムは、ハードウェア内で直接に実行可能である。コンピュータシステム(5)は、異種のマルチプロセッシングシステムを特徴とし得る。エミュレーションコプロセッサ(50、150)が外部アプリケーションプログラムを実行している一方で、ホストプロセッサ(48、152)は、外部アプリケーションプログラムと関連しないオペレーティングシステムルーチンを実行することが可能であり、またはホストアプリケーションプログラムを実行することが可能である。

